

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号  
特開2002-24079  
(P2002-24079A)

(43)公開日 平成14年1月25日(2002.1.25)

|                          |       |               |                              |
|--------------------------|-------|---------------|------------------------------|
| (51)Int.Cl. <sup>7</sup> | 識別記号  | F I           | テーマコード(参考)                   |
| G 0 6 F 12/00            | 5 4 6 | G 0 6 F 12/00 | 5 4 6 T 5 B 0 8 2<br>5 4 6 A |

審査請求 未請求 請求項の数20 O L 外国語出願 (全 79 頁)

|             |                             |         |  |
|-------------|-----------------------------|---------|--|
| (21)出願番号    | 特願2001-129926(P2001-129926) | (71)出願人 | 595124929<br>マイクロソフト コーポレイション<br>MICROSOFT CORPORATI<br>ON<br>アメリカ合衆国 98052-6399 ワシント<br>ン州 レドモンド ワン マイクロソフト<br>ウェイ (番地なし) |
| (22)出願日     | 平成13年4月26日(2001.4.26)       | (72)発明者 | バード、デイリー エス.<br>アメリカ合衆国、98033 ワシントン州、<br>カークランド、エヌイー 103ド ストリ<br>ート 11411  |
| (31)優先権主張番号 | 0 9 / 5 7 3 6 5 6           | (74)代理人 | 100095555<br>弁理士 池内 寛幸 (外3名)   |
| (32)優先日     | 平成12年5月18日(2000.5.18)       |         |  |
| (33)優先権主張国  | 米国 (U S)                    |         |  |

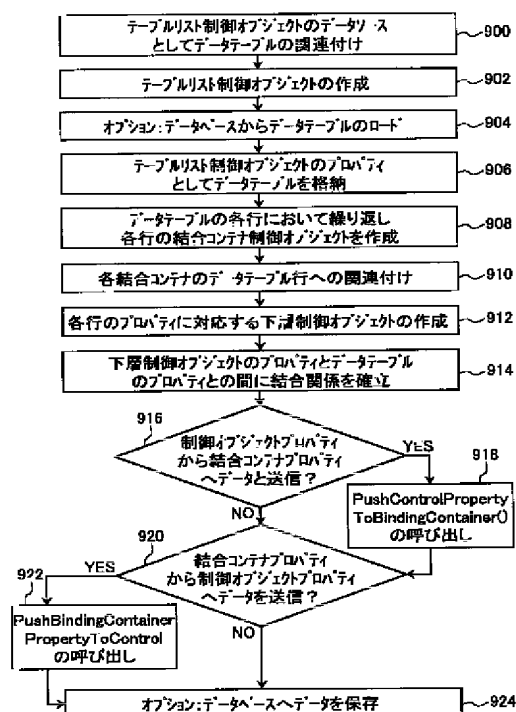
最終頁に続く

(54)【発明の名称】 サーバ側制御オブジェクトを用いるデータ結合

(57)【要約】 (修正有)

【課題】開発者が最小のプログラムでウェブページを動的に作成及び処理できるフレームワークを提供。

【解決手段】制御オブジェクト階層の中に作成される制御オブジェクトのプロパティはサーバ側データソースのプロパティにデータ結合することができて、階層的データ結合関係が、制御オブジェクトのプロパティとデータソースのプロパティとの間に確立される。テンプレート宣言は、データソースのデータオブジェクトに対応する結合コンテナオブジェクトの構成を規定するのに用いられる。繰返し制御オブジェクトは、結合コンテナオブジェクト数に従って増加するデータソースのデータオブジェクト数を判断する。単一のデータ結合のタイプは、(1)データソースから制御オブジェクトへの一方向データ結合、(2)制御オブジェクトからデータソースへの一方向データ結合、(3)制御オブジェクトとデータソース間における二方向データ結合を含む。



## 【特許請求の範囲】

【請求項1】 クライアントコンピュータシステムに接続されたサーバコンピュータにおいて、下層サーバ側制御オブジェクトのプロパティを、1つ以上のデータセットを有するサーバ側データテーブル内のデータセットのプロパティとデータ結合する方法であって、前記下層サーバ側制御オブジェクトがクライアント側ユーザインタフェースに対応しており、前記サーバ側データテーブルの各データセットを繰り返し参照して、各データセットに対応する結合コンテナサーバ側制御オブジェクトを作成する工程と、各結合コンテナサーバ側制御オブジェクトを前記サーバ側データテーブルのデータセットの1つに関連付ける工程と、各データセットのプロパティとして、所与のデータセットに対応する結合コンテナサーバ側制御オブジェクトの下層にあり、かつ前記所与のデータセットのプロパティと関連付けられた下層サーバ側制御オブジェクトを作成する工程と、前記下層サーバ側制御オブジェクトのプロパティと前記サーバ側データテーブルのデータセットのうちの1つのプロパティとの間にデータ結合関係を確立させる工程とを含むデータ結合する方法。

【請求項2】 サーバ側データベースから前記サーバ側データテーブルをロードする工程と、繰り返しサーバ側制御オブジェクトを前記サーバ側データテーブルに関連付ける工程と、前記繰り返しサーバ側制御オブジェクトを、各結合コンテナサーバ側制御オブジェクトが前記繰り返しサーバ側制御オブジェクトの子オブジェクトであるサーバ側制御オブジェクト階層の中に作成する工程と、前記サーバ側データテーブルを前記繰り返しサーバ側制御オブジェクトのプロパティとして格納する工程とをさらに含む請求項1に記載のデータ結合する方法。

【請求項3】 前記繰り返しサーバ側制御オブジェクトを関連付ける工程において、前記サーバ側データテーブルを参照するために前記繰り返しサーバ側制御オブジェクトのデータソースプロパティを設定する工程を含む請求項2に記載のデータ結合する方法。

【請求項4】 前記繰り返し工程において、各結合コンテナサーバ側制御オブジェクトが、前記サーバ側データテーブルの所与のデータセットと関連付ける各結合コンテナサーバ側制御要素を、前記繰り返しサーバ側制御オブジェクトの子として作成する工程を含む請求項2に記載のデータ結合する方法。

【請求項5】 前記確立させる工程において、前記下層サーバ側制御オブジェクトの制御識別子、前記下層サーバ側制御オブジェクトのプロパティの第1のプロパティ識別子、前記結合コンテナサーバ側制御オブジェクトの結合コンテナ識別子、前記サーバ側データテ

ブルのデータセットのプロパティの第2のプロパティ識別子を含む結合オブジェクトを作成する工程を含む請求項1に記載のデータ結合する方法。

【請求項6】 前記データ結合関係に基づいて、前記サーバ側データテーブルのデータセットのプロパティとして格納するために前記下層制御オブジェクトのプロパティとして格納されたデータを送信する工程をさらに含む請求項1に記載のデータ結合する方法。

【請求項7】 前記データ結合関係に基づいて、前記下層制御オブジェクトのプロパティに格納するために前記サーバ側データテーブルのプロパティとして格納されたデータを受信する工程をさらに含む請求項1に記載のデータ結合する方法。

【請求項8】 クライアントコンピュータシステムに接続されているサーバコンピュータにおいて、コンピュータシステムによって搬送波で具体化され、下層サーバ側制御オブジェクトのプロパティを、1つ以上のデータセットを有するサーバ側データテーブル内のデータセットのプロパティとデータ結合するコンピュータプロセスデータを実行処理するコンピュータプログラムをコード化するコンピュータデータ信号であって、前記下層サーバ側制御オブジェクトがクライアント側ユーザインタフェース要素に対応しており、前記コンピュータプロセスは、前記サーバ側データテーブルの各データセットを繰り返し参照して、各データセットに対応する結合コンテナサーバ側制御オブジェクトを作成する工程と、各結合コンテナサーバ側制御オブジェクトを前記サーバ側データテーブルのデータセットの1つに関連付ける工程と、各データセットのプロパティとして、所与のデータセットに対応する結合コンテナサーバ側制御オブジェクトの下層にあり、かつ前記所与のデータセットのプロパティと関連付けられた下層サーバ側制御オブジェクトを作成する工程と、前記下層サーバ側制御オブジェクトのプロパティと前記サーバ側データテーブルのデータセットのうちの1つのプロパティとの間にデータ結合関係を確立する工程とを含むコンピュータデータ信号。

【請求項9】 クライアントコンピュータシステムに接続されたサーバコンピュータにおいて、コンピュータシステムによって読み取り可能であり、下層サーバ側制御オブジェクトのプロパティを、1つ以上のデータセットを有するサーバ側データテーブル内のデータセットのプロパティとデータ結合するコンピュータプロセスデータを実行するコンピュータプログラムをコード化しているコンピュータプログラム記憶媒体であって、前記下層サーバ側制御オブジェクトがクライアント側ユーザインタフェース要素に対応しており、前記コンピュータプロセスは、

前記サーバ側データテーブルの各データセットを繰り返し参照して、各データセットに対応する結合コンテナサーバ側制御オブジェクトを作成する工程と、

各結合コンテナサーバ側制御オブジェクトを前記サーバ側データテーブルのデータセットうちの1つに関連付ける工程と、

各データセットのプロパティとして、所与のデータセットに対応する結合コンテナサーバ側制御オブジェクトの下層にあり、かつ前記所与のデータセットのプロパティと関連付けられた下層サーバ側制御オブジェクトを作成する工程と、

前記下層サーバ側制御オブジェクトのプロパティと前記サーバ側データテーブルのデータセットの1つのプロパティとの間にデータ結合関係を確立する工程とを含むコンピュータプログラム記憶媒体。

【請求項10】 下層サーバ側制御オブジェクトのプロパティを、1つ以上のデータオブジェクトを有するサーバ側データアレイのプロパティとデータ結合するためのコンピュータプロセスをコンピュータシステム上で実行するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって、

前記子サーバ側制御オブジェクトはクライアント側ユーザインタフェース要素に対応し、

前記コンピュータプロセスは、

サーバ側データベースから前記サーバ側データベースアレイをロードする工程と、

繰り返しサーバ側制御オブジェクトを前記サーバ側データテーブルに関連付ける工程と、

前記繰り返しサーバ側制御オブジェクトをサーバ側制御オブジェクト階層内に作成する工程と、

前記サーバ側データアレイを前記繰り返しサーバ側制御オブジェクトのプロパティとして格納する工程と、

前記繰り返しサーバ側制御オブジェクトの子であって、前記データオブジェクトの1つと対応する結合コンテナサーバ側制御オブジェクトを作成する工程と、前記結合コンテナサーバ側制御オブジェクトを前記サーバ側データアレイのデータオブジェクトの1つに関連付ける工程と、

下層サーバ側制御オブジェクトを各データオブジェクトの各プロパティのために作成する工程であって、各下層サーバ側制御オブジェクトが前記結合コンテナサーバ側制御オブジェクトの下層にある作成工程と、

前記下層サーバ側制御オブジェクトのプロパティと前記サーバ側データアレイのプロパティとの間にデータ結合関係を確立する工程とを含むコンピュータプログラムプロダクト。

【請求項11】 前記繰り返しサーバ側制御オブジェクトに関連付ける工程において、

前記サーバ側データアレイを参照するために前記繰り返しサーバ側制御オブジェクトのデータソースプロパティ

を設定する工程を含む請求項10に記載のコンピュータプログラムプロダクト。

【請求項12】 前記繰り返しサーバ側制御オブジェクトに関連付ける前記繰り返し工程において、

前記サーバ側データアレイの所与のデータオブジェクトに関連付けられる各結合コンテナサーバ側制御要素を、前記繰り返しサーバ側制御オブジェクトの子として作成する工程を含む請求項11に記載のコンピュータプログラムプロダクト。

【請求項13】 前記確立する工程において、前記下層サーバ側制御オブジェクトの制御識別子、前記下層サーバ側制御オブジェクトのプロパティのプロパティ識別子、前記結合コンテナサーバ側制御オブジェクトの結合コンテナ識別子、前記サーバ側データアレイのデータオブジェクトのプロパティのプロパティ識別子を含む結合関係オブジェクトを作成する工程を含む請求項10に記載のコンピュータプログラムプロダクト。

【請求項14】 前記データ結合関係に基づいて、前記サーバ側データアレイのデータオブジェクトのプロパティとして格納するために前記下層制御オブジェクトのプロパティとして格納されたデータを送信する工程をさらに含む請求項10に記載のコンピュータプログラムプロダクト。

【請求項15】 前記データ結合関係に基づいて、前記下層制御オブジェクトのプロパティ内に格納するための前記サーバ側データアレイのデータオブジェクトのプロパティとして格納されたデータを受信する工程をさらに含む請求項10に記載のコンピュータプログラムプロダクト。

【請求項16】 クライアント側ユーザインタフェース要素に対応するサーバ側制御オブジェクトの階層を用いてサーバ側データ結合を実行するサーバであって、各データオブジェクトがプロパティを含んでいる、1つ以上のデータオブジェクトを有するサーバ側データアレイと、

サーバ側制御オブジェクト階層内にあり、かつ前記サーバ側データアレイに関連付けられた繰り返しサーバ側制御オブジェクトと、

前記サーバ側データアレイにおける多くのデータオブジェクトに基づいて前記繰り返しサーバ側制御オブジェクトにより繰り返し作成される1つ以上の結合コンテナサーバ側制御オブジェクトと、

各下層サーバ側制御が前記サーバ側データアレイにおいて所与のデータオブジェクトの前記結合制御サーバ側制御オブジェクトの子として作成される、前記サーバ側データアレイの各データオブジェクトのプロパティに対応する1つ以上の下層サーバ側制御オブジェクトと、

前記下層制御オブジェクトのプロパティと前記サーバ側データアレイのデータオブジェクトのプロパティとの間におけるデータ結合関係を記述するデータ結合関係構造

とを含むサーバ。

【請求項17】 前記データ結合関係に基づいて、前記下層制御オブジェクトのプロパティからのデータを、前記サーバ側データアレイのデータオブジェクトのプロパティへ格納するプッシュモジュールをさらに含む請求項16に記載のサーバ。

【請求項18】 前記データ結合関係に基づいて、前記サーバ側データアレイのデータオブジェクトのプロパティからのデータを前記下層制御オブジェクトのプロパティに格納するプッシュモジュールをさらに含む請求項16に記載のサーバ。

【請求項19】 前記サーバ側データアレイにロードされたサーバ側データ格納部の一部をさらに含む請求項16に記載のサーバ。

【請求項20】 前記サーバ側データアレイが保存されるサーバ側データ格納部の一部をさらに含む請求項16に記載のサーバ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般にウェブサーバフレームワークに関し、特にサーバ側データテーブルのプロパティを二方向に結合するサーバ側制御オブジェクトに関する。

【0002】

【従来の技術】典型的なウェブブラウザは、クライアントシステムにおいて表示するウェブページの外観および基本動作を規定するウェブサーバからデータを受け取る。典型的な手順としては、ユーザが、ワールドワイドウェブ上の資源のグローバルアドレスであるユニホームリソース（資源）ロケータ（以下、「URL」という）を特定し、所望のウェブサイトにアクセスする。一般に、用語「資源」とは、プログラムによってアクセスすることができるデータまたはルーチンのことである。URLの一例は、“HYPERLINK”`http://www.microsoft.com/ms.htm` `http://www.microsoft.com/ms.htm`”である。このURL例の第1の部分は、通信に用いる所与のプロトコル（例えば“http”）を示している。第2の部分は、資源の所在を示すドメイン名（例えば“HYPERLINK”`http://www.microsoft.com`” `www.microsoft.com`”）を特定している。第3の部分はドメイン内の資源（例えば“ms.htm”と呼ばれるファイル）を特定している。これに従い、ブラウザは、HYPERLINK”`http://www.microsoft.com`” `www.microsoft.com` ドメイン内のms.htmファイルに関連するデータを取り出すためのURL例に関連するHTTP（ハイパーテキストトランスポートプロトコル）リクエストを生成する。`www.microsoft.com`サイトをホストしているウェブサーバはHTTPリクエストを受け取り、要求されたウェブページまたは資源をクライアントシステムにHTTPレスポンスで戻し、ブラウザに表示する。

【0003】上記の例の“ms.htm”ファイルは、静的なHT

ML（ハイパーテキストマークアップ言語）コードを含んでいる。HTMLは、ワールドワイドウェブ上でのドキュメント（例えば、ウェブページ）作成に用いられるプレインテキストオーサリング言語である。このようなものであるため、HTMLファイルは、ウェブサーバから取り出され、ブラウザにウェブページとして表示し、ユーザがインターネットからの情報を見ている間に期待するようになる豊かなグラフィカルな経験を提供することができる。HTMLを用いて、開発者が、例えば、ブラウザに表示するフォーマット化されたテキスト、リスト、フォーム、テーブル、ハイパーテキストリンク、インライン画像および音声、ならびに背景グラフィックスを特定する。しかし、HTMLファイルは、ウェブページコンテンツの動的な生成を本質的にサポートしていない静的ファイルである。

【0004】株価の変化や交通情報などのような動的コンテンツを表示する場合は、通常、より複雑なクライアント・サーバ対話を扱うためのサーバ側アプリケーションプログラムを開発する。サーバ側アプリケーションプログラムは、HTTPリクエストを処理し、クライアントへのHTTPレスポンスの送信に適切なHTMLコードを生成する。HTTPリクエストの一例は、照会ストリングでのデータまたはウェブベースのフォームからのデータなどのパラメータを含むことがある。このようなものであるため、サーバ側アプリケーションプログラムは、このパラメータを処理し、クライアントへのHTTPレスポンスでHTMLコードを動的に生成することができる。サーバ側アプリケーションプログラムの一例として、メモリ機構への1つ以上のフォーマット化されたテキスト書込み処理のシーケンスを用いて、適切なHTMLコードを含むドキュメントを動的に生成する場合がある。その後、得られたドキュメントは、HTTPレスポンスでクライアントシステムに送信され、そこでウェブページとしてブラウザに表示される。

【0005】サーバ側アプリケーションプログラムの開発は、ウェブページ設計に用いる通常のHTMLコード化に精通しているだけでなく、1つ以上のプログラム言語（例えば、C++、Perl、Visual Basic、またはJavaScript）を含むプログラムベシックスに精通していることが要求される複雑な作業である。しかし、ウェブページ設計者は、グラフィックデザイナーまたはエディタであることが多く、プログラム経験がない場合がある。さらに、複雑なウェブページ開発を単純化すると、いかなる開発者によっても新たなウェブコンテンツの開発の速度を上げることができる。一般に、カスタムサーバ側アプリケーションプログラムの開発もまた、多大な労力が要求され、実際、開発者はしばしばそれを試みたくないと思う程である。したがって、開発者が最小のプログラムでウェブページを動的に作成および処理することができる開発フレームワークを提供することが望ましい。

## 【0006】

【発明が解決しようとする課題】動的ウェブページ生成のプログラム要件を最小にする1つの手段は、マイクロソフト社によって提供されるアクティブサーバページ(ASP)フレームワークである。ASPリソースは、典型的に、所望の資源としてASPリソースを特定するHTTPリクエストを処理し、その後、クライアントへのHTTPレスポンスでの結果HTMLコードを生成する、例えばVisual BasicまたはJscriptを含む。さらに、ASP資源は、所与のアプリケーションプログラム労力を軽減するために、予め開発されたまたは第三者のクライアント側ライブラリコンポーネント(例えば、クライアント側ACTIVEX制御)を参照することができる。しかし、現在のサーバ側アプリケーションフレームワークにおいて、サーバ側アプリケーション内でクライアント側ユーザインタフェース要素(例えば、テキストボックス、リストボックス、ボタン、ハイパーリンク、画像、音声など)を動的に管理することを要求されるプログラミングは、依然として高度なプログラミング技術と相当な労力を必要とする。未解決の課題は、ウェブページ開発者がウェブページの他のアスペクトに焦点を当てることができるように、ユーザインタフェース要素を処理することが要求されるプログラミングを適切にカプセル化することについてである。

【0007】あるクライアント側ユーザインタフェース要素は、データベースのようなサーバ側データ格納部に関連付けられたデータを含む。このような要素の一例は、製品と価格の一覧をサーバ側プロダクトデータベースから表示するクライアント側テーブルである。また別の一例としては、後の利用のために、サーバ側顧客データベースに格納すべき SHIPPING アドレスを消費者が入力できるテキストボックスも含まれる。従来は、クライアント側ユーザインタフェースは、ページ開発者によってページに書き込まれ、ページとサーバ側データベースとの間でデータ通信を行うために、アプリケーションプログラムはアプリケーション開発者によって書かれていた。問題は、ページ開発もアプリケーションプログラム開発もともに、多大な開発労力を要求し、ならびにデータベース構造およびある程度の複雑なプログラミング(例えば、クライアント側のスクリプティングあるいはサーバ側のアプリケーションプログラミングを介して)についての深い知識を必要とする点である。

## 【0008】

【課題を解決するための手段】本発明によると、上述および他の課題は、1つ以上のサーバ側制御オブジェクトと1つ以上のサーバ側データ格納部とを結合する階層データを提供することによって解決される。階層データ結合は、データアレイにおける各データオブジェクトに対し結合コンテナを自動的に作成することによってサポートされる。各結合コンテナは、その子供に対し、関連付けられているサーバ側データ格納部のデータオブジェク

トを含むそのパブリックプロパティへのアクセスを提供する。このようにすることで、サーバ側制御オブジェクトのプロパティは、その結合コンテナを介して、サーバ側データ格納部のプロパティに結合され得る。データ結合関係は、サーバ側制御オブジェクトのプロパティからサーバ側データ格納部のプロパティへ、またはその逆方向に、一方向的に確立することができるが、これに限定されない。データ結合関係はまた、サーバ側制御オブジェクトのプロパティとサーバ側データ格納部のプロパティとの間に二方向的に確立することもできるが、これに限定されない。

【0009】ある本発明にかかる手段において、下層サーバ側制御オブジェクトのプロパティを1つ以上のデータセットを有するサーバ側データテーブルのデータセット(例えば、テーブル行)のプロパティにデータを結合する方法が、クライアントコンピュータシステムに接続しているサーバコンピュータにおいて提供される。この下層サーバ側制御オブジェクトは、クライアント側ユーザインタフェース要素に対応する。繰り返し制御オブジェクトは、サーバ側データテーブルの各データセットごとに繰り返し参照し、各データセットに対応する結合コンテナサーバ側制御オブジェクトを作成する。各結合コンテナサーバ側制御オブジェクトは、サーバ側データテーブルのデータセットに関連づけられている。下層サーバ側制御オブジェクトは、各データセットのプロパティに対し作成される。所与の下層サーバ側制御オブジェクトは、所与のデータセットに対応する結合コンテナサーバ側制御オブジェクトの下層にあり、かつ所与のデータセットのプロパティと関連付けられている。データ結合関係は、下層サーバ側制御オブジェクトのプロパティと、サーバ側データテーブルのデータセットのうちの1つとの間に確立される。

【0010】本発明にかかる他の手段において、製品はコンピュータプログラムプロダクトとして提供される。コンピュータプログラムプロダクトのある実施形態において、コンピュータシステムによって読み取り可能であり、かつ下層サーバ側制御オブジェクトのプロパティを1つ以上のデータオブジェクトを有するサーバ側データアレイのプロパティにデータ結合させるコンピュータプロセスを実行するためのコンピュータプログラムをコード化するコンピュータプログラム記憶媒体が提供される。コンピュータプログラムプロダクトのまた別の実施形態としては、コンピュータシステムによって搬送波で具体化され、かつコンピュータプログラムをコード化するコンピュータデータ信号において提供することもできる。子サーバ側制御オブジェクトは、クライアント側ユーザインタフェース要素に対応する。サーバ側データアレイは、サーバ側データベースからロードされる。繰り返しサーバ側制御オブジェクトは、サーバ側データテーブルに関連付けられ、かつサーバ側制御オブジェクト階

層中に作成される。サーバ側データアレイは、繰り返しサーバ側制御オブジェクトのプロパティとして格納される。データオブジェクトの1つと対応する結合コンテナサーバ側制御オブジェクトは、繰り返しサーバ側制御オブジェクトの子として作成される。結合コンテナサーバ側制御オブジェクトは、サーバ側データアレイのデータオブジェクトのうちの1つに関連付けられる。下層サーバ側制御オブジェクトは、各データオブジェクトの各プロパティに対して作成される。各下層サーバ側制御オブジェクトは、結合コンテナサーバ側制御オブジェクトの下層にある。データ結合関係は、下層サーバ側制御オブジェクトのプロパティとサーバ側データアレイのプロパティとの間に確立される。

【0011】別の手段においては、1つ以上のデータオブジェクトを有するサーバ側データアレイを含む、クライアント側ユーザインタフェース要素に対応するサーバ側制御オブジェクトの階層を用いてサーバ側データ結合を実行するサーバもまた提供される。各データオブジェクトはプロパティを含んでもよい。サーバ側制御オブジェクト階層中の繰り返しサーバ側制御オブジェクトは、サーバ側データアレイに関連付けられている。1つ以上の結合コンテナサーバ側制御オブジェクトは、サーバ側データアレイにおける多数のデータオブジェクトに基づいて、繰り返しサーバ側制御オブジェクトによって繰り返し作成される。1つ以上の下層サーバ側制御オブジェクトは、サーバ側データアレイの各データオブジェクトのプロパティに対応する。各下層サーバ側制御は、サーバ側データアレイにおける所与のデータオブジェクトの結合制御サーバ側制御オブジェクトの下層に作成される。データ結合関係構造は、下層制御オブジェクトのプロパティとサーバ側データアレイのデータオブジェクトのプロパティとのデータ結合関係を記述する。

#### 【0012】

【発明の実施の形態】本発明のある実施形態において、ウェブページコンテンツは、ウェブサーバ上に動的に生成され、クライアントに表示される。クライアントは、例えば、標準HTMLあるいは他のオーサリング言語をサポートするいかなるブラウザにもなることができる。ウェブサーバにおけるクライアントは、HTTPリクエストおよびHTTPレスポンスなどを用いることによって、ネットワークを通じて通信する。このように、ウェブサーバは、おそらくHTMLコード形態であろうが、ウェブページコンテンツを生成し、クライアントへそのコンテンツを送信する。これによりブラウザにウェブページを表示することができる。ウェブページの個々のユーザインタフェース要素と論理的に対応し得るサーバ側制御オブジェクトがウェブサーバ上に作成され、ウェブページコンテンツを処理し、レンダリングする。サーバ側制御オブジェクトは、サーバ側制御オブジェクトの階層を作成するページファクトリによって処理されるASP+リソースなどの

動的コンテンツ資源で宣言される。階層における制御オブジェクトはクライアントから受信したリクエストを協働して処理し、次いで、結果ウェブページコンテンツを生成し、クライアントへ送信し、階層における制御オブジェクトを終了させる。

【0013】本発明のある実施形態において、ページオブジェクトは、制御オブジェクト階層の最上位レベルとして例示することができる。制御オブジェクトでもあるページオブジェクトは、典型的に1つ以上の子制御オブジェクトを含み、各子制御オブジェクトは、それ自体1つ以上の子制御オブジェクトを含み、多数レベルの階層に広がる。ページオブジェクトおよび下層制御オブジェクトは、処理シーケンスを実行し、クライアント側ユーザインタフェース要素に対応するウェブコンテンツを処理および生成する。

【0014】このシーケンスにおける処理の1つは、1つ以上のサーバ側制御オブジェクトとサーバ側データ格納部との通信を制御するデータ結合処理を含む。本発明のある実施形態において、サーバ側データ格納部は、ウェブページのユーザインタフェース要素に関するデータを格納するデータベースである(例えば、図1のデータ格納部131を参照)。特に限定されないが、構成登録、データファイル、データストリームを含む他の非データベース型データ格納部もまた、本発明の範囲内に含まれる。サーバ側制御オブジェクトおよび関連のデータ結合関係を動的コンテンツ資源において宣言することにより、ページ開発者は、サーバ側データ格納部に関連するサーバ側データアレイに対して一方向、および/または二方向データ結合関係を有する制御オブジェクト階層を構成することができる。

【0015】図1は、本発明のある実施形態におけるクライアントに表示するウェブページコンテンツを動的に生成するウェブサーバを示す。クライアント100は、クライアント100の表示装置上におけるウェブページ104を表示するブラウザ102を実行する。クライアント100は、ビデオモニタのような表示装置を有するクライアントコンピュータシステムを含んでもよい。マイクロソフト社によって販売されている"INTERNET EXPLORER"ブラウザは、本発明のある実施形態におけるブラウザ102の一例である。他のブラウザの例として、"NETSCAPE NAVIGATOR"および"MOSAIC"などがあるが、これに限らない。例示したウェブページ104には、4つのリスト行要素を有するテーブルリスト140が組み込まれている。リスト行要素142は、テーブルリスト要素140が含む子ユーザインタフェース要素である。リスト行要素142は、テンプレート宣言(テンプレート宣言の一例として図6Aを参照)によって定義され、ラベル要素144(例えば、OrderIDを表す)および146(例えば、注文製品の量を表す)を含む2つの列と、テーブルリスト要素148を含む1つの列を有している。要素

144、146、および148は、リスト行要素142によって含まれる子ユーザインタフェース要素として宣言される（ラベルおよびテーブルリスト宣言の例として図6Aを参照）。付加リスト行要素150、152および154もまたウェブページ104に示されている。

【0016】図6Aの宣言に従って、ウェブページ104のユーザインタフェース要素に対応するサーバ側制御オブジェクトが2つのサーバ側データアレイ（例えば、MyOrderSystemオブジェクトを通じてサーバ側データベースアクセスに関連するサーバ側データテーブルMyOrdersおよびMyItems）にデータ結合される。一般には、サーバ側データアレイはサーバ側データ格納部と関連し（例えば、格納部からロードされ、かつ／または格納部に格納される）、制御オブジェクト階層内においてインデックスデータセット（例えば、データ行、またはデータオブジェクト）を含む。このようなデータセットは、例えば、あるテーブルデータ行のデータ値を表すことができるプロパティを有する。コラム148は、2つの列、すなわち、（1）結合されたItemsデータテーブルからの行インデックスの列、及び（2）Itemsデータテーブルからラベル（例えば、項目名）を含む3つのリスト行要素の列、の2つを有するサブテーブルリスト要素を表している。

【0017】ブラウザ102は、HTTPレスポンス112においてウェブサーバ116からHTMLコードを受信ことができ、HTMLコードにより記述されたウェブページを表示する。ある実施形態を参照してHTMLについて説明するが、特に制限されるものではなく、SGML (Standard Generalized Markup Language) および、XMLベースのマークアップ言語であって、ポケットベル（登録商標）および携帯電話などの狭帯域無線装置の内容およびユーザインタフェースを特定するように設計されたWML (Wireless Markup Language) を含む他のオーサリング言語も本発明の範囲内であると考えられている。さらに、標準HTML3.2が、本明細書中に主に開示されているが、HTMLのいかなるバージョンも本発明の範囲内に含まれ得る。

【0018】クライアント100とウェブサーバ116との通信は、HTTPリクエスト114およびHTTPレスポンス112の一連の処理を用いて行うことができる。ある実施形態を参照してHTTPを説明するが、特に制限されるものではなく、S-HTTPを含めた他のトランスポートプロトコルも本発明の範囲内であると考えられている。ウェブサーバ116において、HTTPパイプラインモジュール118が、HTTPリクエスト114を受信し、URLを解析し、リクエストを処理する適切なハンドラ120を呼び出す。本発明のある実施形態において、異なるタイプの資源を扱う複数のハンドラ120がウェブサーバ116に備わっている。

【0019】例えば、URLがHTMLファイルのような静的なコンテンツ資源122を特定する場合、ハンドラ12

0は、静的コンテンツ資源122にアクセスし、HTTPパイプライン118を介して静的コンテンツ資源122をHTTPレスポンス112でクライアント100へ送信する。あるいは、本発明のある実施形態において、URLがASP+リソースのような動的コンテンツ資源124を特定する場合、ハンドラ120は、動的コンテンツ資源124にアクセスし、動的コンテンツ資源124のコンテンツを処理し、ウェブページ104用の結果HTMLコードを生成する。本発明のある実施形態において、結果HTMLコードは、標準HTML3.2コードを含む。一般に、動的コンテンツ資源は、クライアントに表示すべきウェブページを記述するオーサリング言語を動的に生成するのに用いることができるサーバ側宣言データ格納部（例えば、ASP+リソース）である。そして、ウェブページ用のHTMLコードは、HTTPパイプライン118を通過し、HTTPレスポンス112でクライアント100へ送られる。

【0020】この処理の間、ハンドラ120はまた、開発労力を簡単化するために予め開発された、あるいは第3者コードのライブラリにアクセスすることができる。このようなライブラリの1つがサーバ側クラス制御ライブラリ126であり、ここから、ハンドラ120は、ユーザインタフェース要素を処理しウェブページに表示する結果HTMLデータを生成するサーバ側制御オブジェクトを例示することができる。本発明のある実施形態において、1つ以上のサーバ側制御オブジェクトは、動的コンテンツファイル124に記述されたウェブページ上に、可視的にまたは隠して、1つ以上のユーザインタフェース要素にマッピングする。

【0021】これに対し、第2のライブラリは、マイクロソフト社からの"ACTIVEX"コンポーネントを含むライブラリのようなクライアント側制御クラスライブラリ128である。"ACTIVEX"制御は、クライアントおよび他のコンポーネントとの対話の仕方において一定の標準に従うCOM（コンポーネントオブジェクトモデル）オブジェクトである。クライアント側"ACTIVEX"制御は、例えば、クライアントに自動的にダウンロードされ、クライアントのウェブブラウザによって実行処理され得るCOMベースのコンポーネントである。サーバ側ACTIVEXコンポーネント（図示せず）は、株価検索アプリケーションまたはデータベースコンポーネントのサーバ側機能性を提供するような多様なサーバ側機能を果たすためにサーバ上で実行され得るCOMベースのコンポーネントである。ACTIVEXについては、「ACTIVEXおよびOLEの理解」（デビッド・チャペル、マイクロソフトプレス、1996年）により詳細に記載されている。

【0022】"ACTIVEX"制御とは対照的に、動的コンテンツ資源124に特定される本発明の実施形態のサーバ側制御オブジェクトは、クライアント上のウェブページに組み込まれるユーザインタフェース要素に論理的に対応する。サーバ側制御オブジェクトはまた、例えば、HT

MLタグと所与のクライアント側"ACTIVE"制御をリファレンスするロケータを含み得る有効なHTMLコードを生成することができる。ブラウザが、すでに格納システム内にクライアント側"ACTIVE"制御用コードを有している場合は、クライアント上のウェブページ内で"ACTIVE"制御を実行処理する。そうでなければ、ブラウザは、ロケータによって特定された資源から"ACTIVE"制御用コードをダウンロードし、そして、クライアント上のウェブページ内で"ACTIVE"制御を実行処理する。本発明の実施形態におけるサーバ側制御オブジェクトはまた、サーバ上で株価検索アプリケーションを実行するのに用いられるサーバ側"ACTIVE"制御に対しイベントを立ち上げることができる。

【0023】ハンドラ120はまた、ウェブサーバ116上または別のアクセス可能ウェブサーバ上で実行処理する1つ以上の非ユーザインタフェースサーバコンポーネント130にアクセスする。株価検索アプリケーションまたはデータベースコンポーネントのような非ユーザインタフェースサーバコンポーネント130は、ハンドラ120によって処理される動的コンテンツ資源124において参照され、またはそれに関連付けられる。動的コンテンツ資源124で宣言された制御オブジェクトによって立ち上げられたサーバ側イベントは、非ユーザインタフェースコンポーネント130の適切なメソッドを呼び出すサーバ側コードによって処理され得る。その結果、サーバ側制御オブジェクトによって提供される処理は、ウェブページのユーザインタフェース要素の処理および生成をカプセル化することによって非ユーザインタフェースサーバコンポーネント130のプログラミングを簡単にし、これにより、非ユーザインタフェースサーバコンポーネント130の開発者は、ユーザインタフェース問題ではなく、アプリケーション固有の機能の開発に集中することができる。

【0024】本発明のある実施形態において、サーバ側制御オブジェクトのプロパティは、サーバ側データ格納部131内のプロパティとデータ結合されている。例えば、データ結合によって、ページ開発者はサーバ側制御オブジェクト階層内の制御オブジェクトのプロパティをサーバ側データベース内のデータ項目に関連付けることが可能になる。この処理の間、各制御オブジェクトは、データベース内のデータオブジェクトから関連するプロパティ値を取り出すことによって、そのデータ結合プロパティを更新する（サーバ側データ格納部からの一方向データ結合という）。あるいはまた、各制御オブジェクトは、制御オブジェクト内的プロパティ値を用いて、サーバ側データ格納部のプロパティを更新する（サーバ側データ格納部への一方向データ結合という）。この2つのタイプの一方向データ結合を組み合わせ、サーバ側制御オブジェクトとサーバ側データ格納部との間に二方向データ結合を提供することができる。

【0025】図2は、本発明のある実施形態におけるサーバ側制御オブジェクトを用いてのクライアント側ユーザインタフェース要素の処理および生成処理のフローチャートを示す。送信処理200において、クライアントは、HTTPリクエストをサーバに送信する。HTTPリクエストは、ASP+リソースなどの資源を特定するURLを含む。受信処理202において、サーバは、HTTPリクエストを受信し、特定された資源を処理する適切なハンドラを呼び出す。ASP+リソースは、構文解析処理203で読み出される。作成処理204は、特定された動的コンテンツ資源（例えば、ASP+リソース）の内容に基づいてサーバ側制御オブジェクト階層を生成する。

【0026】処理206において、制御オブジェクト階層のサーバ側制御オブジェクトは、ポストバックイベントハンドリング、ポストバックデータハンドリング、状態管理およびデータ結合のうちの1つ以上の処理を行う。ユーザインタフェース要素からのポストバックイベントおよびデータ（まとめて「ポストバック入力」）は、クライアントからサーバへと送られ処理される。ポストバックイベントは、特に制限されるものではなく、例えば、クライアント側ボタン要素からの「マウスクリック」イベントまたはサーバに送られるクライアント側テキストボックス要素からの「データ変化」イベントを含んでもよい。ポストバックデータは、例えば、これに限らないが、テキストボックス要素またはドロップダウンボックスから選ばれた項目のインデックスにユーザによって入力されたテキストを含んでもよい。

【0027】処理208において、階層内の各サーバ側制御オブジェクトは、クライアント側ユーザインタフェース要素のウェブページでの表示のためのHTMLコードのようなデータを生成（またはレンダリング）するために呼び出される。用語「レンダリング」は、ユーザインタフェース上にグラフィックスを表示する処理を意味することがあるが、本明細書において「レンダリング」は、表示およびクライアント側機能のためのブラウザのようなクライアントアプリケーションによって解釈され得るオーサリング言語データの生成処理をも意味している。処理206およびレンダリング処理208のより詳細な説明は図6との関連で行われている。ある実施形態において、個々の制御オブジェクトにおけるrender()メソッドの呼び出しは、ツリートラバーサルシーケンスを用いて行われる。すなわち、ページオブジェクトのrender()メソッドの呼び出しは、階層内の適切なサーバ側制御オブジェクトにわたる繰り返したラバースになる。適切な制御オブジェクトのrender()メソッドを呼び出す別の方法として、イベントシグナリングまたはオブジェクト登録手法などの方法を用いてもよい。括弧は、データ値と比較するなどのメソッドを示す「render()」ラベルを指定する。

【0028】本発明のある実施形態において、個々のサ



サーバ側制御オブジェクトの実際の作成は、サーバ側制御オブジェクトが処理206または208においてアクセスされる（ポストバック入力のハンドリング、状態のロード、制御オブジェクトからHTMLコードのレンダリングなど）まで遅らせてもよい。サーバ側制御オブジェクトが所与のリクエストのためにアクセスされることがない場合、制御オブジェクトの作成が遅れ、不要な制御オブジェクト作成処理を排除することによって、サーバ処理が最適化される。

【0029】送信処理210において、HTMLコードをHTTPレスポンスでクライアントに送信する。受信処理214において、クライアントは、表示されるべき新たなウェブページに関連するHTMLコードを受信する。表示処理216において、クライアントシステムは、HTTPレスポンスから受け取ったHTMLコードに応じて新しいページのユーザインタフェース要素を組み入れる（例えば、表示する）。しかし、ユーザインタフェース要素の組み入れは、音声または触覚出力の提供、メモリへの読出しおよび書込み、スクリプト処理の制御などの非表示処理を含んでもよいことを理解すべきである。終了処理212において、サーバ側制御オブジェクト階層を終了する。本発明のある実施形態において、階層内サーバ側制御オブジェクトは、関連付けられたASP+リソースを参照するHTTPリクエストに回答して作成され、オーサリング言語データ（例えば、HTMLデータ）のレンダリングが終わると破壊される。別の実施形態において、処理212は、処理208の後、処理210の前に行ってもよい。

【0030】図3は、本発明のある実施形態において用いるウェブサーバでのモジュールの一例を示す。ウェブサーバ300は、HTTPパイプライン304にHTTPリクエスト302を受け入れる。HTTPパイプライン304は、ウェブページ統計のロギング、ユーザ照合、ユーザアクセス権およびウェブページの出力キャッシュ化用モジュールなどの種々のモジュールを含んでもよい。ウェブサーバ300によって受信される各入力HTTPリクエスト302は、最終的には、IHTTPハンドラクラス（ハンドラ306として図示）の特定のインスタンスによって処理される。ハンドラ306は、URLリクエストを解析し適切なハンドラファクトリ（例えば、ページファクトリモジュール308）を呼び出す。

【0031】図3において、ASP+リソース310に関連付けられたページファクトリモジュール308が呼び出され、ASP+リソース310のオブジェクトの例示および構成をハンドリングする。ある実施形態において、ASP+リソースは、ファイルに特定の接尾語（または“.aspx”のようなファイル拡張部分）を指定することによって認識され得る。所与のASP+リソースに対するリクエストがまず、ページファクトリモジュール308によって受信されると、ページファクトリモジュール308は、ファイルシステムを検索して適切なファイル（例えば、.asp

xファイル310）を得る。このファイルは、リクエストを処理するサーバによって後に解釈またはアクセスされ得るテキスト（例えば、オーサリング言語データ）または別のフォーマットでのデータ（例えば、バイトコードデータまたはコード化されたデータ）を含んでもよい。物理的なファイルが存在する場合、ページファクトリモジュール308は、ファイルを開き、そのファイルをメモリに読み出す。ファイルが見つからない場合、ページファクトリモジュール308は、適切な「ファイルが見つからない」というエラーメッセージを戻す。

【0032】ASP+リソース310をメモリに読み出した後、ページファクトリモジュール308は、ファイル内容を処理し、ページのデータモデル（例えば、スクリプトブロックのリスト、ディレクティブ、静的テキスト領域、階層サーバ側制御オブジェクト、サーバ側制御プロパティなど）を構築する。データモデルは、ページベースのクラスを拡張させるCOM+（Component Object Model）クラスのような新たなオブジェクトクラスのソースリストを生成するのに用いられる。ページベースクラスは、基本ページオブジェクトの構造、プロパティおよび機能を定義するコードを含んでいる。本発明のある実施形態において、次いで、ソースリストは、中間言語に動的にコンパイルされ、のちにプラットフォームに固有の命令（例えば、X86, Alphaなど）に適時に（Just-In-Time）コンパイルされる。中間言語は、COM+ IL コード、Java バイトコード、Modula 3 コード、SmallTalk コードおよびVisual Basicコードなどの汎用またはカスタム指向言語コードを含んでもよい。別の実施形態において、中間言語処理を省き、固有の命令をソースリストまたはソースファイル（例えば、ASP+リソース310）から直接生成してもよい。制御クラスライブラリ312は、制御オブジェクト階層の生成に用いられる予め定義されたサーバ側制御クラスを得るためにページファクトリモジュール308によってアクセスされ得る。

【0033】ページファクトリモジュール308は、図1のウェブページ104に相当するサーバ側制御オブジェクトであるページオブジェクト314を出力する。ページオブジェクト314およびその子オブジェクトが、制御オブジェクト階層316の一例である。他の制御オブジェクトの例もまた、本発明に従い考えることができ、カスタム制御オブジェクトと同様に、特に制限されるものではなく、表1のHTML制御に対応するオブジェクトも含んでいる。ページオブジェクト314は、図1のウェブページ104に論理的に対応し、サーバ上の他の制御オブジェクトと階層的に関連している。ある実施形態において、ページオブジェクトは、その子制御オブジェクトを階層的に含んでいるコンテナオブジェクトである。別の実施形態では、依存関係などの他の形態の階層関係を用いることができる。多数レベルの子オブジェクトを有するより複雑な制御オブジェクト階層において、

1つの子オブジェクトが、他の子オブジェクトのコンテンツオブジェクトであってもよい(例えば、テーブルリスト140は、リスト行要素142、150、152および154とその子要素のコンテンツである)。これらのサーバ側制御オブジェクトは、協働してHTTPリクエスト302からの入力をハンドリングし、サーバ側制御オブジェクトの状態を管理し、データ結合を行い、クライアントの結果ウェブページのためにオーサリング言語データ(例えば、HTML)をレンダリングする。結果HTMLコードは、制御オブジェクト階層316からの出力であり、HTTPレスポンス324でクライアントに送信される。

【0034】上記の実施形態において、制御オブジェクト階層316の制御オブジェクトは、サーバ300上で作成および実行され、各サーバ側制御オブジェクトは、クライアント上のユーザインタフェース要素と論理的に対応する。サーバ側制御オブジェクトはまた、協働して、HTTPリクエスト302からのポストバック入力をハンドリングし、サーバ側制御オブジェクトの状態を管理し、サーバ側データベースとのデータ結合を行い、クライアントでの結果ウェブページの表示に用いるオーサリング言語データ(例えば、HTMLコード)を生成する。結果オーサリング言語データは、サーバ側制御オブジェクト階層316から生成(すなわち、レンダリング)され、HTTPレスポンス324でクライアントに送信される。例えば、結果HTMLコードは、いかなる有効なHTML構成も具現化することができ、ACTIVE Xタイプの制御、J A V A (登録商標) アプレット、スクリプトおよびその他、ブラウザによって処理されたとき、クライアント側ユーザインタフェース要素(例えば、制御ボタン、テキストボックスなど)を生み出すいかなるウェブ資源も参照することができる。

【0035】ASP+リソース310でなされた宣言によって、サーバ側制御オブジェクトは、非ユーザインタフェースサーバコンポーネント330とクライアント側ユーザインタフェース要素との対話のために、1つ以上の非ユーザインタフェースサーバコンポーネント330にアクセスすることができる。例えば、ポストバック入力に応答して、サーバ側制御オブジェクトは、サーバ側イベントをそれらのイベント用に登録された非ユーザインタフェースサーバコンポーネントに対し立ち上げることができる。このように、非ユーザインタフェースサーバコンポーネント330は、ユーザとの対話を、ユーザインタフェース要素を介して、これらの要素を表示および処理するのに必要なコードをプログラミングすることなく行うことができる。

【0036】1つ以上のサーバ側データ格納部331のプロパティは、サーバ側制御オブジェクトのプロパティにデータ結合することができる。データ結合の一形態をサポートするよう構成されたサーバ側制御オブジェクト階層の一例についての詳細な説明は、図7に関連して行

われる。このデータ結合関係は、ASP+リソースにおいて宣言され、サーバ側制御オブジェクトをコンパイルし、ランタイム実行する間に確立される。

【0037】要するに、本発明の実施形態は、クライアントに送信するHTMLコードを生成するためにサーバ上で作成および実行処理されるサーバ側制御オブジェクトを含む。HTMLコードは、例えば、いかなる有効なHTML構成も具現化することができ、ACTIVE Xタイプの制御、J A V A アプレット、スクリプトならびにクライアントでのユーザインタフェースボタンおよび他のユーザインタフェース要素を生成する他のいかなるウェブ資源も参照することができる。クライアントでのユーザは、サーバ側制御オブジェクトに論理的に対応するこれらのユーザインタフェース要素と対話し、リクエストをサーバに送り返すことができる。サーバ側制御オブジェクトは、サーバ上で再作成され、クライアントにレスポンスとして送信すべき次のラウンドのHTMLコードを生成するように、ユーザインタフェース要素のデータ、イベントおよび他の特性を処理する。

【0038】図4を参照すると、本発明の実施形態のコンピュータシステムの一例は、プロセッサユニット402、システムメモリ404およびシステムメモリ404を含む種々のシステムコンポーネントをプロセッサユニット400に接続するシステムバス406を含んでいる従来のコンピュータシステム400という形態の汎用コンピュータ装置を含んでいる。システムバス406は、メモリバスまたはメモリコントローラ、種々のバスアーキテクチャを用いるペリフェラルバスおよびローカルバスを含む幾つかのタイプのバス構造のいずれであってもよい。システムメモリは、再生専用メモリ(ROM)408およびランダムアクセスメモリ(RAM)410を含んでいる。コンピュータシステム400内の要素間での情報の転送を助ける基本ルーチンを含んでいる基本入力/出力システム412(BIOS)は、ROM408に格納されている。

【0039】コンピュータシステム400は、さらに、ハードディスクの読み出しおよび書き込みを行うハードディスクドライブ412、着脱可能な磁気ディスク416の読み出しおよび書き込みを行う磁気ディスクドライブ414およびCD-ROM、DVDまたは他の光学媒体のような着脱可能な光ディスク419の読み出しおよび書き込みを行う光ディスクドライブ418を含んでいる。ハードディスクドライブ412、磁気ディスクドライブ414および光学ディスクドライブ418は、それぞれ、ハードディスクドライブインタフェース420、磁気ディスクドライブインタフェース422および光ディスクドライブインタフェース424によってシステムバス406に接続されている。ドライブおよびその関連コンピュータ読み取り可能媒体が、コンピュータシステム400のコンピュータ読み取り可能命令、データ構造、プログラムおよ

び他のデータの不揮発性記憶部を提供している。

【0040】本明細書に記載の上記環境例では、ハードディスク、着脱可能な磁気ディスク416および着脱可能な光ディスク419を用いているが、データ保存可能な他のタイプのコンピュータ読み取り可能媒体を上記システム例に用いることができる。上記動作環境例に用いることができるこれらの他のタイプのコンピュータ読み取り可能媒体は、例えば、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、ベルヌイ (Bernoulli) カートリッジ、ランダムアクセスメモリ (RAM) および再生専用メモリ (ROM) などがある。

【0041】多数のプログラムモジュールが、ハードディスク、磁気ディスク416、光ディスク419、ROM408またはRAM410に格納され得、これらは、オペレーティングシステム426、1つ以上のアプリケーションプログラム428、他のプログラムモジュール430およびプログラムデータ432を含む。ユーザは、コマンドおよび情報をコンピュータシステム400にキーボード434およびマウス436または他のポインティング装置などの入力装置によって入力することができる。他の入力装置としては、例えば、マイクロフォン、ジョイスティック、ゲームパッド、サテライトディッシュおよびスキャナなどがある。これらおよび他の入力装置は、システムバス406に接続されているシリアルポートインタフェース440を介して処理装置402に接続されていることが多い。しかし、これらの入力装置はまた、パラレルポート、ゲームポートまたはユニバーサルシリアルバス (USB) などの他のインタフェースによって接続されていてもよい。モニタ442または他のタイプの表示装置もまた、ビデオアダプタ444などのインタフェースを介してシステムバス406と接続している。モニタ442に加えて、コンピュータシステムは、典型的には、スピーカおよびプリンタなどの他の周辺出力装置 (図示せず) を含む。

【0042】コンピュータシステム400は、リモートコンピュータ446のような1つ以上のリモートコンピュータへの論理接続を用いたネットワーク化された環境で動作することができる。リモートコンピュータ446は、コンピュータシステム、サーバ、ルータ、ネットワークPC、ピア (peer) 装置、または他の共通ネットワークノードであり得、典型的にコンピュータシステム400との関連で上述した要素の多くまたはすべてを含む。ネットワーク接続は、ローカルエリアネットワーク (LAN) 448およびワイドエリアネットワーク (WAN) 450を含む。このようなネットワーク環境は、オフィス、企業規模コンピュータネットワーク、イントラネットおよびインターネットにおいて珍しいものではない。

【0043】LANネットワーク環境で用いるとき、コンピュータシステム400は、ネットワークインタフェースまたはアダプタ452を介してローカルネットワーク

448に接続される。WANネットワーク環境で用いるとき、コンピュータシステム400は、典型的に、インターネットのようなワイドエリアネットワーク450による通信を確立するためのモデム454または他の手段を含む。モデム454は内蔵または外付けのいずれでもよく、シリアルポートインタフェース440を介してシステムバス406と接続されている。ネットワーク化された環境において、コンピュータシステム400に関連して述べたプログラムモジュールまたはその一部は、リモートメモリ記憶装置に記憶されてもよい。図示されたネットワーク接続は例であって、コンピュータ間の通信リンク確立のために他の手段を用いることができる。

【0044】本発明の実施形態において、コンピュータ400は、ウェブサーバを表し、プロセッサ402が、記憶媒体416、412、414、418、419またはメモリ404のうち少なくとも1つに記憶されたASP+リソース上でページファクトリモジュールを実行処理する。HTTPレスポンスおよびリクエストは、クライアントコンピュータ446に接続されたLAN448により通信される。

【0045】図5は、本発明のある実施形態におけるページオブジェクトおよび他の制御オブジェクトのサーバ側処理を表すプロセスフローチャートである。処理500において、ページオブジェクト構築部が、ページファクトリモジュール308によって呼び出される (図3参照)。その結果、ページオブジェクト (例えば、図3におけるページオブジェクト314を参照) は、クライアント上のウェブページユーザインタフェース要素に論理的に対応するように作成される。処理502において、ページファクトリモジュールは、クライアントから受け取ったHTTPリクエストの段階的な処理を始めるページオブジェクトのProcessRequestメンバ関数を呼び出す。本発明の一実施形態の第1の段階において、サーバ側作成処理 (図示せず) は、ページオブジェクトの制御オブジェクト階層に含まれる下層サーバ側制御オブジェクトを作成する。すなわち、子制御オブジェクトの構築部がHTTPリクエスト処理の継続期間中、制御オブジェクトを作成するために繰り返し呼び出される。

【0046】しかし、別の実施形態において、子制御オブジェクトの作成は、制御オブジェクトが所与の処理ステップ (例えば、ポストバックイベントのハンドリング、ポストバックデータのハンドリング、ビューステートのローディングおよび保存、データ結合の解析または対応するユーザインタフェース要素のHTMLコードのレンダリング) に必要とされるまで遅らせることができる。「制御オブジェクト作成を遅らせる」後者の実施形態は、不必要なCPUおよびメモリの利用を減らすことができるので最適である。例えば、クライアントから受け取ったユーザ入力イベントが、全く異なるウェブページの作成ということになる場合がある。この場合、直ちに制

御オブジェクト階層を終了させ、新たなページの新たな異なる制御オブジェクト階層を例示することになるイベントを処理するためだけに、以前のページの制御オブジェクト階層全体を例示する必要はない。

【0047】ページオブジェクトのProcessRequestメソッドのサーバ呼び出しにตอบสนองして、処理504～520は、所与のHTTPリクエストのデータなどに応じ、ページオブジェクトおよび個々の下層制御オブジェクトによって実行処理することができる。本発明のある実施形態において処理504～520を図5の順序で各個々のオブジェクトに対し行う。しかし、1つのオブジェクトに対する所与の処理は、HTTPリクエストによっては、別のオブジェクトの所与の処理に関して順番に行われなかったり、全く処理が行われないこともある。例えば、第1のオブジェクトは、初期化処理504およびそのロード処理506を行い、ポストバックデータ処理508を開始し、その後、遅れた制御オブジェクト作成により下層制御オブジェクトが、それ自体の初期化処理504およびロード処理506を行う。ページオブジェクトおよび下層制御オブジェクトによる処理の順番は、これには限らないが、HTTPリクエストにおけるデータの性質、制御オブジェクト階層の構成、制御オブジェクトの現在の状態および制御オブジェクト作成が遅れたか否かに依存する。

【0048】初期化処理504は、動的コンテンツ資源における初期化に関する任意のサーバ側コードを実行処理することによって制御オブジェクトが作成された後、制御オブジェクトを初期化する。このようにして、各サーバ側制御オブジェクトは、動的コンテンツ資源で宣言された特定のサーバ側機能でカスタマイズされ得る。本発明のある実施形態において、動的コンテンツコードは、サーバ上のASP+リソースでページ開発者によって宣言されたベースページ制御クラスをカスタマイズまたは拡張するものである。ASP+リソースがコンパイルされると、宣言されたコードは、適切な初期コード（例えば、ページオブジェクトおよび下層制御オブジェクトのInit()メソッド）に含まれている。初期化処理504は、このコードを実行処理して、ページベースクラスおよび下層制御オブジェクトのベースクラスをカスタマイズまたは拡張する。

【0049】本発明のある実施形態において、サーバ側制御オブジェクトの状態管理は、サーバ側制御オブジェクトを以前の状態に戻すことによってクライアント・サーバシステムの無状態モデルを収納するために搬送可能状態構造を用いるロード処理506および保存処理516においてサポートされる。ある実施形態では、状態は、一対のHTTPリクエスト／レスポンスの隠れた1つ以上のHTMLフィールドでまたな別の搬送可能状態構造でサーバを往復するが、他の搬送可能状態構造も本発明の範囲内であると考えられる。

【0050】クライアントとサーバ間の現在のページに関する所与のリクエストおよびレスポンスによる一連の処理において、1つ以上の制御オブジェクトの状態は、先のリクエストの処理後に保存処理516によって搬送可能状態構造に記録される。本発明のある実施形態において、階層情報またはサーバが適切な制御オブジェクトと所与の状態とを関連付けることを可能にする制御オブジェクト識別子を含む追加の状態情報もまた、搬送可能状態構造に含まれる。次のHTTPリクエストにおいて、状態情報は、搬送可能状態構造でサーバに戻される。サーバは、受信した搬送可能状態構造から状態情報を抽出し、状態データを制御オブジェクト階層内の適切な制御オブジェクトにロードし、各制御オブジェクトを先のHTTPレスポンス以前に存在したような状態に戻す。現在のリクエストに対する処理後、1つ以上のサーバ側制御オブジェクトの状態は、再度保存処理516によって搬送可能状態構造に記録され、次のHTTPレスポンスで搬送可能状態構造をクライアントに戻す。

【0051】ロード処理506の結果、各サーバ側制御オブジェクトを先のHTTPリクエスト以前の状態と同じ状態にする。例えば、テキストボックス制御オブジェクトが、先のHTTPレスポンス以前に"JDoe"と同等のプロパティ値を含む場合、テキストストリング"JDoe"をそのプロパティ値にロードすることなどによって、ロード処理506は、同じ制御オブジェクトを先の状態に戻す。さらに、所与のオブジェクトの状態が記憶され回復されたかどうかについても構成可能である。

【0052】本発明のある実施形態を要約すると、1つ以上のサーバ側制御オブジェクトの状態が処理後「保存」される。保存された状態情報は、レスポンスによりクライアントに送信される。クライアントは、次のレスポンスで保存された状態情報をサーバ側に戻す。サーバは、階層の状態が以前の状態に戻るように、新たに例示されたサーバ側制御オブジェクト階層に状態情報をロードする。

【0053】別の実施形態では、サーバ上、またはサーバからクライアント、そしてサーバへ戻るという往復の間、サーバによってアクセス可能な幾つかの他のウェブロケーションに状態情報を保持できる。クライアントリクエストをサーバが受信した後、この状態情報は、サーバによって取り出され、制御オブジェクト階層内の適切なサーバ側制御オブジェクトにロードされ得る。

【0054】処理508において、HTTPリクエストから受け取ったポストバックデータが処理される。ポストバックデータは、対になったキー値、階層表示（例えば、XML）またはRDF (Resource Description Framework) のような他のデータ表示でのHTTPリクエストのペイロードに含まれ得る。処理508は、ペイロードを構文解析してターゲットサーバ側制御オブジェクトの一意の識別子を識別する。識別子（例えば、"page:tablelist1:listr

ow2:label1")を見つけ、ターゲットサーバ側制御オブジェクトが制御オブジェクト階層に存在する場合、対応するポストバックデータがその制御オブジェクトへと渡される。例えば、図1を参照すると、リスト行142および列144のデータ項目に関連する一意の識別子および対応データは、HTTPリクエスト114のペイロードでウェブサーバ116へ送られる。処理508はHTTPリクエスト114のペイロードを構文解析し、テーブルリスト140の一意の識別子とその関連値（すなわち、リスト行142のようなユーザインタフェース要素のHTMLコード）を得る。それから処理508は、リスト行142の一意の識別子を解析し、対応するサーバ側制御オブジェクトを識別およびトラバースし、処理するターゲット制御オブジェクトにポストバック値を渡す。

【0055】ロード処理506に関して説明したように、サーバ側制御オブジェクトのプロパティ値は、以前の状態に戻され得る。ポストバックデータを受け取ると、サーバ側制御オブジェクトは、渡されたポストバック値が対応する先のプロパティ値を変化させたかどうかを判断する。変化した場合には、関連制御オブジェクトのデータ変化を示す変化リストにその変化をロギングする。すべてのポストバックデータが制御オブジェクト階層内で処理された後、制御オブジェクトメソッドが呼び出され、サーバ上で起動している株価検索アプリケーションのような1つ以上の非ユーザインタフェースサーバコンポーネントに対し、1つ以上のポストデータ変化イベントを立ち上げ得る。ポストバックデータ変化イベントは、例えば、ポストバックデータがサーバ側制御オブジェクトのプロパティを変化させたということを示しているイベントである。例示的な実施形態において、このようなイベントは、システム提供イベント待ち行列に送られ、イベントを処理するよう登録されたサーバ側コードを呼び出すことができる。次いで、サーバ側コードは、非ユーザインタフェースサーバコンポーネントのメソッドを呼び出すことができる。このように、サーバ側非ユーザインタフェースサーバコンポーネントは、サーバ側制御オブジェクトのデータの変化によってトリガされたイベントに応答することができる。アプリケーション提供イベント待ち行列、ポーリング、および処理割込みを用いることを含む、イベントを実行する別の方法もまた本発明の範囲内と考えることができる。

【0056】処理510において、ポストバックイベントをハンドリングする。ポストバックイベントは、HTTPリクエストのペイロードで通信される。処理510は、そのイベントが向けられているサーバ側制御オブジェクトを識別する特定のイベントターゲット（例えば、本発明のある実施形態では、"#EVENTTARGET"とラベル付けされている）を構文解析する。さらに、処理510は、探し当てたイベント引数がある場合はそれを構文解析し、イベント引数（例えば、本発明のある実施形態では、"#

EVENTARGUMENT"とラベル付けされている）を特定されたサーバ側制御オブジェクトに与える。制御オブジェクトは、動的コンテンツ資源に関連する非ユーザインタフェースサーバコンポーネントによって処理するイベントを立ち上げる。

【0057】処理512は、サーバ側制御オブジェクトとサーバがアクセス可能な1つ以上のサーバ側データベースとのデータ結合関係を解明し、これにより、データベース値で制御オブジェクトプロパティを更新し、かつ／または、制御オブジェクトプロパティの値でデータベースフィールドを更新する。本発明のある実施形態では、サーバ側制御オブジェクトのプロパティは、サーバ側アプリケーションデータベースの表のような親データ結合コンテナのプロパティと関連付けられ（またはデータ結合され）得る。データ結合処理512の間に、ページフレームワークは、対応する親データ結合コンテナプロパティの値を有するデータ結合された制御オブジェクトプロパティを更新することができる。このように、次のレスポンスにおけるウェブページ上のユーザインタフェース要素は、更新されたプロパティ値を正確に反映する。なぜなら、ユーザインタフェース要素が対応する制御オブジェクトプロパティは、データ結合処理512の間に、自動的に更新されたからである。同様に、制御オブジェクトプロパティはまた、親データ結合コンテナフィールドに更新され得、これにより、サーバ側制御オブジェクトからのポストバック入力でサーバ側アプリケーションデータベースを更新する。

【0058】所与のサーバ制御オブジェクトと所与のデータソース（例えば、サーバ側データテーブル）とのデータ結合関係は、ASP+リソースにおいて宣言される（例えば、図6Aのライン14を参照）。ウェブページにおいて結合データの共通の階層的性質をサポートするために、データ結合関係はそれ自体の階層をサポートすることができる（図7の説明を参照）。制御オブジェクト階層において、データ結合関係は、データテーブルおよびサーバ側制御オブジェクトのプロパティが結合されているデータテーブルのプロパティを識別する。データ結合関係はまた二方向であり得るが、これは、（1）サーバ側データテーブルのプロパティ値がサーバ側制御オブジェクトの結合プロパティにロードされ得、かつ（2）サーバ側制御オブジェクトのプロパティ値がサーバ側データテーブルの結合プロパティにロードされ得るということを意味している。データ結合処理512の後、すべてのサーバ側制御オブジェクトは、サーバ側データテーブルから適切に更新されたデータ結合プロパティ値を正確に反映する。

【0059】処理514は、制御オブジェクト状態が保存され出力がレンダリングされる以前に実行処理され得る多数の更新処理を行う。処理516は、制御オブジェクト階層内の1つ以上の制御オブジェクトから状態情報

(すなわち、ビューステート)を要求し、状態情報を格納し、HTTPレスポンスペイロードでクライアントへ送られる搬送可能状態構造に挿入する。例えば、“grid”制御オブジェクトが、次のHTTPリクエスト(すなわち、処理506)の後この状態に戻ることができるように、

“grid”制御オブジェクトは、値のリストの現在のインデックスページを保存する。上記のように、ビューステート情報は、クライアントによる次のアクション以前の制御オブジェクト階層の状態を表す。ビューステート情報が戻ってくると、それは、制御オブジェクト階層を、クライアントポストバック入力処理またはデータ結合以前の先の状態にするのに用いられる。

【0060】レンダリング処理518は、HTTPレスポンスでクライアントへ送信する適切なオーサリング言語出力(例えば、HTMLデータ)を生成する。レンダリングは、すべてのサーバ側制御オブジェクトのトップダウン階層ツリーウォークおよび埋め込まれたレンダリングコードにより行われる。処理520は、任意の最終のクリーンアップ作業(例えば、ファイルを閉じたりデータベースの接続)を行い、制御オブジェクト階層を終了させる。次いで、処理は502に戻り、処理522を行い、そこでページオブジェクトは、その破壊部を呼び出すことによって終了する。

【0061】図6Aは、本発明の実施形態においてサーバにデータ結合制御オブジェクトを宣言する動的コンテンツ資源(例えば、ASP+リソース)の一例を示す。ライン1は、HTMLファイルの開始タグである<html>をリテラルとして含む。ライン2から8までは、ローカルの“MyOrders”および“MyItems”のデータアレイがポピュレートするコードのコード宣言ブロックを表す。具体的には、ライン2は、“runat=server”属性および値によってサーバ側のコードをサーバ側スクリプトとして宣言する。このスクリプトは、ページオブジェクトの図5のロード処理506を無効にする。ライン3から5は、サーバ側制御オブジェクトのネームスペース内でアクセス可能なパブリックオブジェクトを宣言する。ライン3は、パブリックオブジェクトである“OrderSystem”タイプの“MyOrderSystem”を宣言し、これがライン7で使用され、ライン4においてパブリックデータアレイオブジェクトの“MyOrders”として宣言されたデータアレイをサーバ側データ格納部から取り出す。ライン8において、パブリックオブジェクトの“MyOrderSystem”はまた、ライン5におけるパブリックデータアレイオブジェクトの“MyItems”として宣言されたデータアレイをサーバ側データ格納部から取り出すのにも用いられる。

【0062】ライン11は、HTMLファイルの本文の開始タグ<body>であるリテラルを含む。ライン12は、ウェブページ上に表示をするヘッダ“H1”を規定するリテラルHTMLコードを含む。ライン13は、サーバ側フォーム制御オブジェクトの宣言を含む。ライン14は、“Page.M

yOrders”に等しいデータ結合データソースプロパティを有するサーバ側テーブルリスト制御オブジェクトを宣言する。したがって、MyOrdersデータテーブルは、テーブルリスト制御オブジェクトおよびその下層のデータソースとして宣言されるのである。

【0063】ライン15から24のインラインテンプレート宣言は、ライン14で宣言されたテーブルリスト制御オブジェクトの1つ以上の子制御オブジェクトを宣言する。テンプレート宣言は、構文解析時に解釈されて、適切なITemplateインタフェースインスタンスを繰り返し制御オブジェクト(例えば、テーブルリスト制御オブジェクト)における同名のプロパティへ動的に結合する。テーブルリスト制御オブジェクト(すなわち、繰り返し制御オブジェクト)は、実行時にITemplateインスタンスを使用して、適切なテンプレート制御クラスの新たな0からNのインスタンスを作成および初期化する。作成された新たなインスタンスの数は、データソースの対応するデータセット(例えば、データ行)の数によって制御することができる。

【0064】例えば、ライン15から24は、ライン14において宣言されたテーブルリスト制御オブジェクトのリスト行制御オブジェクトに対し、“itemtemplate”というテンプレートを宣言する。テンプレート制御オブジェクトのパラメータはライン16から22において宣言され、OrderIDラベル制御オブジェクト、Quantityラベル制御オブジェクト、およびその他のテーブルリスト制御オブジェクトを含むリスト行制御オブジェクトを示す。ライン14において宣言されたテーブルリスト制御オブジェクトは、データテーブルMyOrdersのデータセットを繰り返し参照し、各データセットに対して下層リスト行制御オブジェクトを作成する。

【0065】itemtemplateテンプレートに対応するリスト行制御オブジェクトは、制御オブジェクト開発者によってIBindingContainerインスタンスとして開発され、したがって“結合コンテナ”と見なされる。図5のロード処理506の間に、MyOrdersデータテーブルはライン7のコードによってMyOrderSystemからポピュレートされる。例えば、MyOrdersSystemオブジェクトにより提供されたデータは、注文および製品データを含むサーバ側データベースから抽出される。その後、テーブルリスト制御オブジェクトは、データテーブルMyOrdersの各データセットに対し結合コンテナとしてリスト行制御オブジェクトを作成する。

【0066】ライン16において宣言されたOrderID制御オブジェクトは、MyOrdersデータテーブル内のデータセットのOrderIDプロパティに結合されている。OrderIDラベル制御オブジェクトのPropertyBindingオブジェクトによって定義された、OrderIDラベル制御オブジェクトとリスト行制御オブジェクトとの階層データ結合関係は、ライン20において宣言されたラベル制御オブジェ

クトのTextプロパティが、"OrderID"として識別されるデータ項目プロパティに結合されているデータであることを確立する。階層データ結合関係はまた、データ項目プロパティは制御オブジェクトの結合コンテナ、すなわちMyOrdersデータテーブルのプロパティであることを特定する。結合されてOrderID制御オブジェクトとなるMyOrdersデータテーブルのプロパティはデータ項目ではあるが、サーバ側データアレイ（例えば、データテーブルあるいは別のタイプのデータオブジェクト）のいかなるパブリックプロパティも、行インデックス、データ値サイズ、データ値タイプなどを含むサーバ側制御オブジェクトのプロパティに結合することができる。

【0067】ライン17において宣言されたQuantity制御オブジェクトは、MyOrdersデータテーブルにおけるデータセットのQuantityプロパティに結合されている。Quantityラベル制御オブジェクトのPropertyBindingオブジェクトによって定義された、Quantityラベル制御オブジェクトとリスト行制御オブジェクトとの階層データ結合関係は、ライン17において宣言されたラベル制御オブジェクトのTextプロパティが、"Quantity"として識別されるデータ項目プロパティに結合されているデータであることを確立する。階層データ結合関係はまた、データ項目プロパティがQuantityラベル制御オブジェクトの結合コンテナのプロパティ、すなわちMyOrdersデータテーブルのプロパティであることを特定する。結合されてQuantityラベル制御オブジェクトとなるMyOrdersデータテーブルのプロパティはデータ項目ではあるが、サーバ側データアレイ（例えば、データテーブルあるいは別のタイプのデータオブジェクト）のいかなるパブリックプロパティも、データセットインデックス、データ値サイズ、データ値タイプなどを含むサーバ側制御オブジェクトのプロパティに結合されてもよい。

【0068】ライン18は、"Page.MyItems"に等しいデータ結合データソースプロパティを有するサーバ側テーブルリスト制御オブジェクトの宣言を含む。したがって、MyItemsは、テーブルリスト制御オブジェクトおよびその下層のデータソースとして宣言される。ライン19から22のインラインテンプレート宣言は、ライン18において宣言されたテーブルリスト制御オブジェクトの1つ以上の子制御オブジェクトを宣言する。テーブルリスト制御オブジェクト（すなわち、別の繰り返し制御オブジェクト）は、インラインテンプレート宣言に関連するテンプレートインスタンスを用いて、適切なテンプレート制御クラスの0からMの新たなインスタンスを作成および初期化する。作成された新たなインスタンスの数は、データソースにおける対応するデータセットの数によって制御することができる。

【0069】テンプレート制御オブジェクトのパラメータは、Indexラベル（すなわち、所与のデータセットの順序インデックスに対応する）およびデータソースから

データ項目プロパティのテキストを表示するItemラベルを含むリスト行制御オブジェクトを示しているライン20から21において定義される。データテーブルMyItemsの各データセットに対し、ライン18で宣言されたテーブルリスト制御オブジェクト下でリスト行制御オブジェクトが作成される。ファイル600における残りのラインは、ファイル内の入れ子の宣言を閉じる終了タグを含む。

【0070】ASP+リソース600は、本発明のある実施形態において、一方向データ結合（すなわち、サーバ側データテーブルから制御オブジェクト階層へデータを送信すること）を実行するための宣言およびサーバ側スクリプティングを示す。また別の実施形態においては、反対方向の一方向データ結合（すなわち、制御オブジェクト階層からサーバ側データテーブルへデータを送信すること）あるいは二方向データ結合（すなわち、両方向にデータを送信すること）を宣言することができる。

【0071】第1のタイプの一方向データ結合（すなわち、制御オブジェクト方向へ）を実行するために、ライン7におけるコードはLoad()サブルーチンを無効にし、例えば、MyOrderSystemデータテーブルからの注文のスナップショットデータアレイをローカルのMyOrdersデータアレイ（すなわち、上述の第1のタイプの一方向データ結合）へロードする。その後、データ結合段階の間に、サーバ側制御オブジェクトのプロパティはライン14、16、17、18、20および21のデータ結合ステートメントに従って、ローカルのMyOrdersデータアレイからの値から更新することができる。

【0072】第2のタイプの一方向データ結合（すなわち、制御オブジェクトから）を実行するために、ページ開発者はページオブジェクトのSave()サブルーチンを無効にすることができる（図5の保存処理516を参照）。それ以前に、データ結合段階において、ローカルのMyOrdersおよびMyItemsデータアレイは、ライン14、16、17、18、20および21のデータ結合ステートメントに従って、サーバ側制御オブジェクトプロパティからの値を用いて更新されている。その後、無効にされたSave()サブルーチンは、ローカルデータアレイの値をMyOrderSystemオブジェクトのデータテーブルにロードする。ページオブジェクトのSave()サブルーチンを無効にするコード宣言ブロックの一例は、図6BのASP+リソース602におけるライン6から9に示されている。このタイプの一方向データ結合は、例えば、注文をサーバ側データベースにロードするのに用いることができる。二方向データ結合を宣言するために、ページ開発者はページオブジェクトのLoad()およびSave()サブルーチンの両方を無効にすることができる。

【0073】図7は、本発明の一実施形態において、図6Aの動的コンテンツ資源の一例（例えば、ASP+リソース）を処理した結果の制御オブジェクト階層の一部を示

したものである。最上レベルのページ制御オブジェクトおよび種々のリテラル制御オブジェクトは、制御オブジェクト階層700から省略されている。図6Aのライン11から開始するフォーム宣言に対応するフォーム制御オブジェクト702はテーブルリスト制御オブジェクト704を含む。テーブルリスト制御オブジェクト704は、図6Aのライン12から開始して宣言され、MyOrdersデータテーブルに等しいデータ結合プロパティを有することになる。

【0074】テーブルリスト制御オブジェクト704は、リスト行制御オブジェクト706から708（すなわち、ListRow0からListRowNまで）を含む。このレベルでの制御オブジェクト階層におけるリスト行制御オブジェクトの実際の数は、MyOrdersデータテーブル内のデータセットの数に依存している。各リスト行制御オブジェクトの構成は、図6Aのライン13で開始する項目テンプレート宣言によって定義される。各リスト行制御オブジェクトのデータ結合関係は、各リスト行制御オブジェクトの結合コンテナ（すなわち、テーブルリスト制御オブジェクト704）のデータソースに関連して定義される。ある実施形態において、制御オブジェクト結合コンテナは、IBindingContainerインタフェースをサポートする制御オブジェクト階層においてもっとも近いところにあるより高いレベルの制御オブジェクトである。

【0075】図6Aのライン14および15の宣言に対応しているラベル制御オブジェクト710および712は、結合コンテナリスト行オブジェクト706に含まれている。ラベル制御オブジェクト710および712のデータ結合関係は、MyOrdersデータテーブルのデータセットにデータ結合している結合コンテナリスト行制御オブジェクト706に関連して定義される。

【0076】図6Aのライン16における宣言に対応するテーブルリスト制御オブジェクト714もまた、結合コンテナリスト行制御オブジェクト706に含まれている。テーブルリスト制御オブジェクト714は、サブデータテーブルのMyOrders、MyIngredients、に結合したデータとして宣言される結合コンテナである。テーブルリスト制御オブジェクト714は、サブデータテーブルMyOrders、MyIngredients、のデータセットに対応しているリスト行制御オブジェクト716から718（ListRow0からListRowNまで）を含む。リスト行制御オブジェクト716から718までの各々は、図6Aのライン19から開始するテンプレート宣言によって定義される。リスト行制御オブジェクト716から718のデータ結合関係は、結合コンテナテーブルリスト制御オブジェクト714に関連して定義される。

【0077】リスト行制御オブジェクト716から718までの各々は、ラベル制御オブジェクト（例えば、720および722）を含む。例えば、ラベル制御オブジェクト720は、図6Aのライン20においてサブデー

タテーブルMyOrders、MyIngredientsのデータセットのインデックスフィールドにデータ結合したものと宣言され、かつラベル制御オブジェクト722は、図6Aのライン21においてサブデータテーブルMyOrders、MyIngredientsのIngredientsデータ項目にデータ結合したものと宣言される。Ingredientsデータ項目のデータソースを判断するために、ラベル制御オブジェクト720および722のプロパティは、より高いレベルの階層内においてもっとも近い結合コンテナ（すなわち、リスト行制御オブジェクト716のような、IBindingContainerインスタンスである制御オブジェクト）へのリファレンスを戻す。

【0078】リスト行制御オブジェクト708以下の階層は、同一のテンプレート宣言によって定義されるため、リスト行制御オブジェクト706以下の階層と同じ構成に従う。この階層は、ラベル制御オブジェクト730、ラベル制御オブジェクト734およびテーブルリスト制御オブジェクト732を含む第1のレベルを含んでいる。テーブルリスト制御オブジェクト732より下の、他のレベルの階層には、0からNのリスト行制御オブジェクト736から738が含まれる。リスト行制御オブジェクト736から738の各々は、対応するテンプレート宣言によって定義されるようなラベル制御オブジェクトを含む（ラベル制御オブジェクト742、744、746および748を参照）。

【0079】図8は、本発明の一実施形態におけるサーバ側制御クラスの一例の表記を表している。サーバ側制御クラスは、本発明のある実施形態におけるすべてのサーバ側制御オブジェクトに共通のメソッド、プロパティおよびイベントを定義している。より具体的な制御クラス（例えば、ウェブページでのクライアント側ボタンに対応するサーバ側ボタン制御オブジェクト）は、この制御クラスから派生している。制御クラス800は、プロパティ802およびメソッド804を格納するメモリを含むものとして示されている。データメンバとメソッドとの組み合わせが異なる他の制御クラスの実施形態もまた、本発明の範囲内で考えられる。

【0080】示された実施形態において、プロパティ802およびメソッド804はパブリックである。プロパティ"ID"は、制御オブジェクト識別子を示している読み取りおよび書き込み可能なストリング値である。プロパティ"Visible"は、対応しているクライアント側ユーザインタフェース要素のオーサリング言語データをレンダリングすべきかどうかを示す読み取りおよび書き込み可能なブーリアン値である。プロパティ"MaintainState"は、制御オブジェクトが、現在のページリクエストの終了時に（すなわち、図5の保存処理516へのレスポンスで）そのビューステート（およびその子オブジェクトのビューステート）を保存すべきかどうかを示している読み取りおよび書き込み可能なブーリアン値である。プ



ロパティ"Parent"は、制御オブジェクト階層の現在の制御オブジェクトに関連する制御コンテナオブジェクトへの読み取り可能なリファレンスである。プロパティ"Page"は、現在の制御オブジェクトがホストされているルートページオブジェクトへの読み取り可能なリファレンスである。プロパティ"Form"は、現在の制御オブジェクトがホストされているフォーム制御オブジェクトへの読み取り可能なリファレンスである。プロパティ"Trace"は、開発者のトレースログの書きこみを可能にするトレースコンテキストへの読み取り可能なリファレンスである。プロパティ"BindingContainer"は、制御オブジェクトの直接データ結合コンテナへの読み取り可能なリファレンスである。プロパティ"Bindings"は、制御オブジェクトのデータ結合連関の集合へのリファレンスである。

【0081】メソッド804は、リクエストの処理および制御オブジェクトのデータメンバへのアクセス方法を含む。ある実施形態において、メソッドは、制御オブジェクトのメモリスペースに格納されるポインタによって参照される。この参照技術はまた、コンテナ制御オブジェクト、制御コレクションオブジェクト、およびページオブジェクトを含む他のサーバ側オブジェクトの実施形態においても用いられる。メソッド"Init()"は、子制御オブジェクトが作成された後、これらを初期化するのに用いられる(図5の処理504を参照)。メソッド"Load()"は、先のHTTPリクエストからビューステート情報を回復するのに用いられる(図5の処理506を参照)。メソッド"Save()"は、後のHTTPリクエストで用いるビューステート情報を保存するのに用いられる(図5の処理516を参照)。メソッド"PreRender()"は、ビューステートの保存およびコンテンツのレンダリングに先立って必要なプレレンダリングステップを行うのに用いられる(図5の処理514を参照)。メソッド"Render(TextWriter output)"は、現在の制御オブジェクトに対応するユーザインタフェース要素のオーサリング言語コードを出力するのに用いられる(図5の処理518を参照)。コードは、"Output"パラメータでこのコードに渡されたTextWriter Output Screenに出力される。メソッド"Dispose()"は、制御オブジェクトを終了する前に最終的なクリーンアップ作業を行うのに用いられる(図5の処理520を参照)。

【0082】メソッド"GetUniqueID()"は、現在の制御オブジェクトの一意の、階層的に認定されたストリング識別子を得る。メソッド"GetControlWithID(String id)"は、与えられた識別子("id")を有する直接の子制御オブジェクトへのリファレンスを戻す。メソッド"GetControlWithUniqueID(String id)"は、一意の階層識別子("id")を有する子制御オブジェクトへのリファレンスを戻す。

【0083】メソッド"PushControlPropertyTwoBindingContainer(String prop Name)"は、プッシュモジュール

の一例であり、ポストバック値がサーバ側制御オブジェクト内で変化すると、ポストバックデータから二方向のデータ結合をする結合コンテナを更新するのに用いられる。メソッド"PushBindingContainer PropertyTwoControl(String prop Name)"は、別のプッシュモジュールの一例であり、現在の結合コンテナ値を有するサーバ側制御オブジェクトプロパティを更新するのに用いられる。メソッド"HasBindings()"は、制御オブジェクトが結合連関を有しているかどうかを示しているブーリアン値を戻す。これらのメソッドは、サーバ側制御オブジェクトのプロパティとサーバ側データ格納部に関連しているサーバ側データアレイにおけるプロパティとの結合関係を解明するのに用いられる(図5のデータ結合処理512を参照)。

【0084】図9は、本発明のある実施形態における、サーバ側制御オブジェクトのプロパティをサーバ側データアレイのプロパティにデータ結合させる処理のフローチャートを示す。図9に示された処理は、図5に示した1つ以上の処理において実行することができる。例えば、本発明のある実施形態においては、処理900および902は、作成処理または初期化処理504の間に実行し、かつ、処理904はロード処理506の間に実行し、その結果データテーブルは処理906におけるテーブルリスト制御オブジェクトのプロパティとして格納される。ポストバック入力プロセス処理508および510、またはデータ結合処理512を行なっている間にテーブルリスト制御オブジェクトあるいはその子制御オブジェクトがアクセスされる時は、データテーブルの構成に従って、処理908、910、912および914が実行されて結合コンテナおよびその子を含むように階層が拡張される。処理916、918、920および922は、図5のデータ結合処理512の間に実行し、かつ、処理924は図5の保存処理の間に実行する。上述の処理対応関係は、本発明のある実施形態に用いられているが、他の実施形態では、図5および9の処理の順序を入れ替えたり並べ替えたりすることができる。

【0085】図9の個々の処理については、ページがサーバ側データベースからの一方向データ結合をサポートしている場合、あるいはサーバ側データベースと制御オブジェクト階層との間に二方向データ結合をサポートしている場合、処理900は、COM+リフレクションの機能を用いて、サーバ側データベースの一部からローカルデータテーブル(すなわち、データアレイ)をロードする。処理902は、データテーブルをテーブルリスト制御オブジェクトのデータソースとして関連付ける。ある実施形態において、この関連付けは、テーブルリスト制御オブジェクトの宣言において特定されたデータ結合プロパティによって遂行される(図6Aのライン14を参照)。処理904は、制御オブジェクト階層中にテーブルリスト制御オブジェクトを作成する。処理906は、

データテーブルをテーブルリスト制御オブジェクトのプロパティとして格納する。

【0086】処理908において、テーブルリスト制御オブジェクトはデータテーブルの各データセットに対し繰り返し参照する。探知した各データセットに対し、関連するテンプレート宣言に基づいて、テーブルリスト制御オブジェクトは結合コンテナ制御オブジェクト(例えば、リスト行制御オブジェクト)を作成する。各結合コンテナ制御オブジェクトは、テーブルリスト制御オブジェクトの子として扱われる。処理910は、データソース(たとえば、“MyOrders.Row(1)”)のデータセットと等しくなるように結合コンテナ制御オブジェクトのデータ結合プロパティを設定することにより、各結合コンテナ制御オブジェクトをデータテーブルのデータセットに関連させる。

【0087】処理912において、各結合コンテナ制御オブジェクトは、データソースの結合コンテナ対応データセットのプロパティに対応する下層制御オブジェクトを作成する。あるいは、各結合コンテナ制御オブジェクトは子制御オブジェクトを作成するが、それは最終的に、データソースの結合コンテナ対応データセットのプロパティに対応する下層制御オブジェクトを作成することになる。処理914は、データソースの対応データセットのプロパティにおいて、下層制御オブジェクトのプロパティ間の結合関係を確立する。この結合関係は、下層制御オブジェクトのプロパティに格納されたProperty Bindingオブジェクトによって定義することができる。

【0088】決定処理916は、制御オブジェクトプロパティからのデータが、対応する結合コンテナプロパティに送信されるべきかどうかを探知する。送信されるべきと探知した場合、処理918は下層制御オブジェクト内のPushControl PropertyToBindingContainer()メソッドを呼び出して、データを結合コンテナに送信する。その後、手順は決定処理920に進む。あるいはまた、決定処理916が、制御オブジェクトプロパティからのデータは対応する結合コンテナプロパティに送信されるべきと判断しない場合もまた、手順は決定処理916から決定処理920へと進み、決定処理920が、データを結合コンテナプロパティから対応する制御オブジェクトプロパティに送信すべきかどうかを判断する。送信されるべきと探知した場合、処理932は下層制御オブジェクト内のPushBindingContainerPropertyToControl()メソッドを呼び出して、結合コンテナからデータを受信する。その後、処理924に進む。あるいは、決定処理920が、結合コンテナプロパティからのデータは制御オブジェクトプロパティに送信すべきでないと判断した場合もまた、手順は決定処理920から決定処理924へ進む。このページが、サーバ側データベースへの一方データ結合あるいはサーバ側データベースと制御オブジェクト階層との間の二方向データ結合をサポートする場

合、処理924は、COM+リフレクションによって、ローカルなデータテーブルをサーバ側データ格納部の一部へ保存する。次いで、手順は図5のレンダリング処理518へと続く。

【0089】図10は、本発明のある実施形態における、サーバ側プロパティ結合コレクションクラスの一例の表記を示す。サーバ側プロパティ結合コレクションクラスは、プロパティ結合コレクションオブジェクトのメソッドとプロパティとを定義する(図8のサーバ側制御クラスの結合プロパティを参照)。プロパティ結合コレクションクラス1000は、プロパティ1000およびメソッド1004に格納するメモリを含むものとして示されている。データメンバとメソッドとの組み合わせが異なる他のプロパティ結合コレクションクラスの実施形態もまた、本発明の範囲内であると考えられる。

【0090】表示された実施形態において、プロパティ1002および方法1004はパブリックである。プロパティ“All”は、プロパティ結合オブジェクトの読み取りおよび書き取り可能なスナップショットアレイであり、インデックスにより順序付けて、コレクションにおけるすべてのプロパティ結合を表す。プロパティ“this[int index]”は、インデックスで順序付けられたコレクション内のプロパティ結合オブジェクトへのリファレンスを戻す読み取り可能なプロパティ結合オブジェクトである。プロパティ“Count”は、コレクション内のプロパティ結合オブジェクトの数を示す読み取り可能な整数値である。

【0091】メソッド1004は、プロパティ結合コレクションオブジェクトのデータにアクセスするための方法を含む。メソッド“Add(PropertyBinding value)”は、特定のプロパティ結合オブジェクトをコレクションに追加するのに用いられる。メソッド“IndexOf(PropertyBinding value)”は、コレクション内のプロパティ結合オブジェクトの順序インデックスを得るために用いられる。メソッド“GetEnumerator(bool AllowRemove)”は、コレクション内のすべてのプロパティ結合のエニューメレータ(enumerator)を得るのに用いられる。メソッド“Remove(PropertyBinding value)”は、特定のプロパティ結合オブジェクトをコレクションから除去するのに用いられる。メソッド“Remove(int index)”は、インデックスにより特定されたプロパティ結合オブジェクトをコレクションから除去するのに用いられる。メソッド“Clear()”は、コレクションからすべてのプロパティ結合オブジェクトを除去するのに用いられる。

【0092】図11は、本発明のある実施形態における、サーバ側プロパティ結合クラスの一例の表記を示す。サーバ側プロパティ結合クラスは、プロパティ結合オブジェクトのメソッドおよびプロパティを定義し(図11のプロパティ結合コレクションクラスの“All”および“this[int index]”のプロパティを参照)、これにより

制御オブジェクトとサーバ側データソースとの間のデータ結合関係を定義する。プロパティ結合クラス1100は、プロパティ1102を格納するメモリを含むものとして示される。データメンバおよびメソッドの組み合わせが異なる他のクラスの実施形態もまた、本発明の範囲内であると考えられる。

【0093】例示した実施形態において、プロパティ1102はパブリックである。プロパティ"BindingContainer"は、読み取りおよび書き込み可能な結合コンテナインタフェース（すなわち、IBindingContainerオブジェクト）であり、結合コンテナデータソースを識別し、そこへのアクセスを得るのに用いられる。IBindingContainerオブジェクトは、マーカインタフェースであり、自身のパブリックプロパティをその下層の結合に利用させる制御を識別する。したがって、"BindingContainer"プロパティは、結合された制御オブジェクトが結合先のサーバ側データソースを探し出しアクセスを可能にする結合コンテナ識別子を提供する。プロパティ"Property"は、結合コンテナオブジェクト内の結合されたプロパティを識別する読み取りおよび書き込み可能なストリング値（すなわち、プロパティ識別子）である。プロパティ"Control"は、読み取りおよび書き込み可能な制御オブジェクトである（すなわち、データソースプロパティが結合されるターゲットの制御オブジェクトを識別する）。プロパティ"ControlProperty"は、ターゲット制御オブジェクトのターゲットプロパティを識別する読み取りおよび書き込み可能なストリング値（すなわち、プロパティ識別子）である。プロパティ"FormatString"は、読み取りおよび書き込み可能なストリング値であり、オプションフォーマット集中ストリングを識別する。

【0094】本明細書中の本発明の実施形態は、1つ以上のコンピュータシステムの論理ステップとして実行される。本発明の論理処理は、(1) 1つ以上のコンピュータシステムで実行処理されるプロセッサ実行ステップシーケンスとして、(2) 1つ以上のコンピュータシステム内の相互接続された機械モジュールとして実行される。実行は選択の問題であり、本発明を実行するコンピュータシステムの性能要件に依存する。したがって、本明細書に記載の本発明の実施形態を構成する論理処理は、処理、ステップ、オブジェクトまたはモジュールと様々に表現することができる。

【0095】上記の明細書、実施例およびデータは、本発明の実施形態の構造および使用の完全な説明を提供している。本発明は、本発明の精神および範囲から逸脱することなく多数の形態で実施することができるので、本発明は添付の請求項にある。

【0096】

【発明の効果】

【図面の簡単な説明】

【図1】 本発明のある実施形態におけるクライアントに表示するウェブページコンテンツを動的に生成するウェブサーバを示す。

【図2】 本発明のある実施形態におけるサーバ側制御オブジェクトを用いてクライアント側ユーザインタフェース要素の処理およびレンダリングのための処理のフローチャートを示す。

【図3】 本発明のある実施形態で用いるウェブサーバのモジュールの一例を示す。

【図4】 本発明のある実施形態を実行するのに有用なシステムの一例を示す。

【図5】 本発明のある実施形態におけるページオブジェクトの処理を表すプロセスフローチャートを示す。

【図6A】 本発明のある実施形態におけるデータ結合制御オブジェクトを宣言する動的コンテンツ資源（例えば、ASP+リソース）の一例を示す。

【図6B】 本発明のある実施形態におけるサーバ側データテーブルへの第2のタイプの一方向データ結合を宣言するコード宣言ブロックの一例を示す。

【図7】 本発明のある実施形態における図6Aの動的コンテンツ資源（例えば、ASP+リソース）例の処理の結果生ずる制御オブジェクト階層の一部を示す。

【図8】 本発明のある実施形態におけるサーバ側制御クラスの一例の表記を示す。

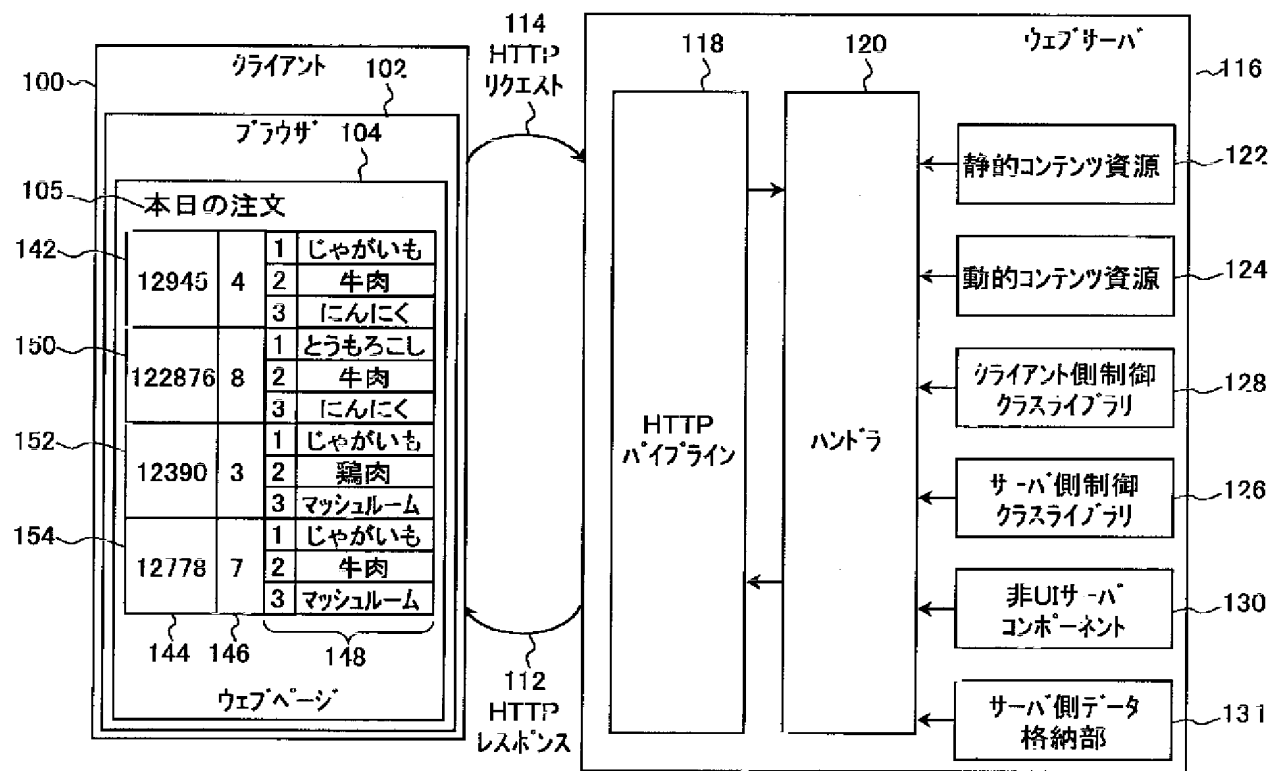
【図9】 本発明のある実施形態におけるサーバ側制御オブジェクトのプロパティとサーバ側データアレイのプロパティとを結合させるデータ処理のフローチャートを示す。

【図10】 本発明のある実施形態におけるサーバ側プロパティ結合コレクションクラスの一例の表記を示す。

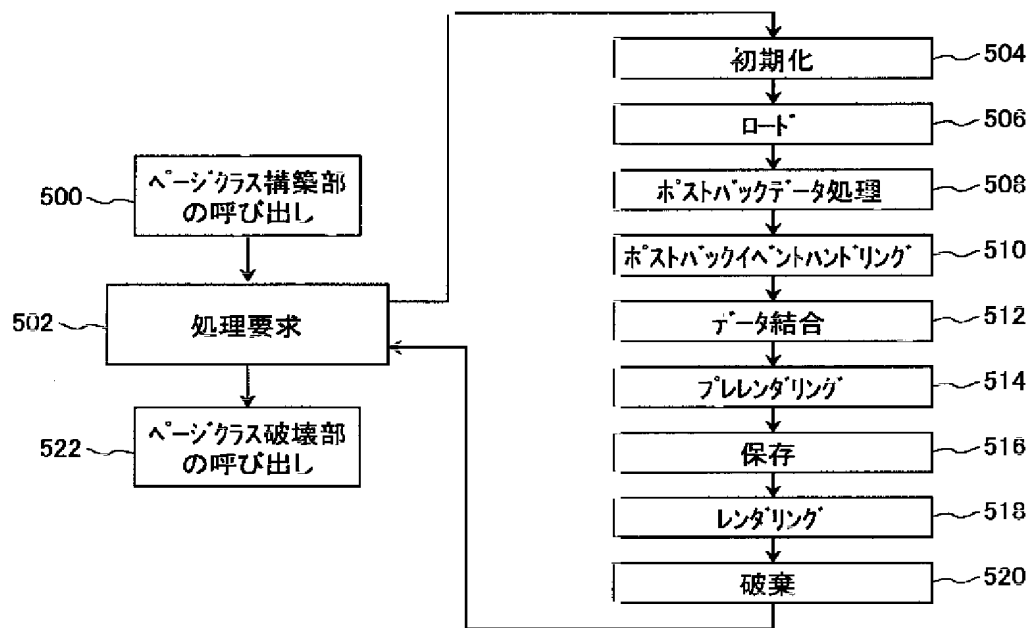
【図11】 本発明のある実施形態におけるサーバ側プロパティ結合クラスの一例の表記を示す。

【符号の説明】

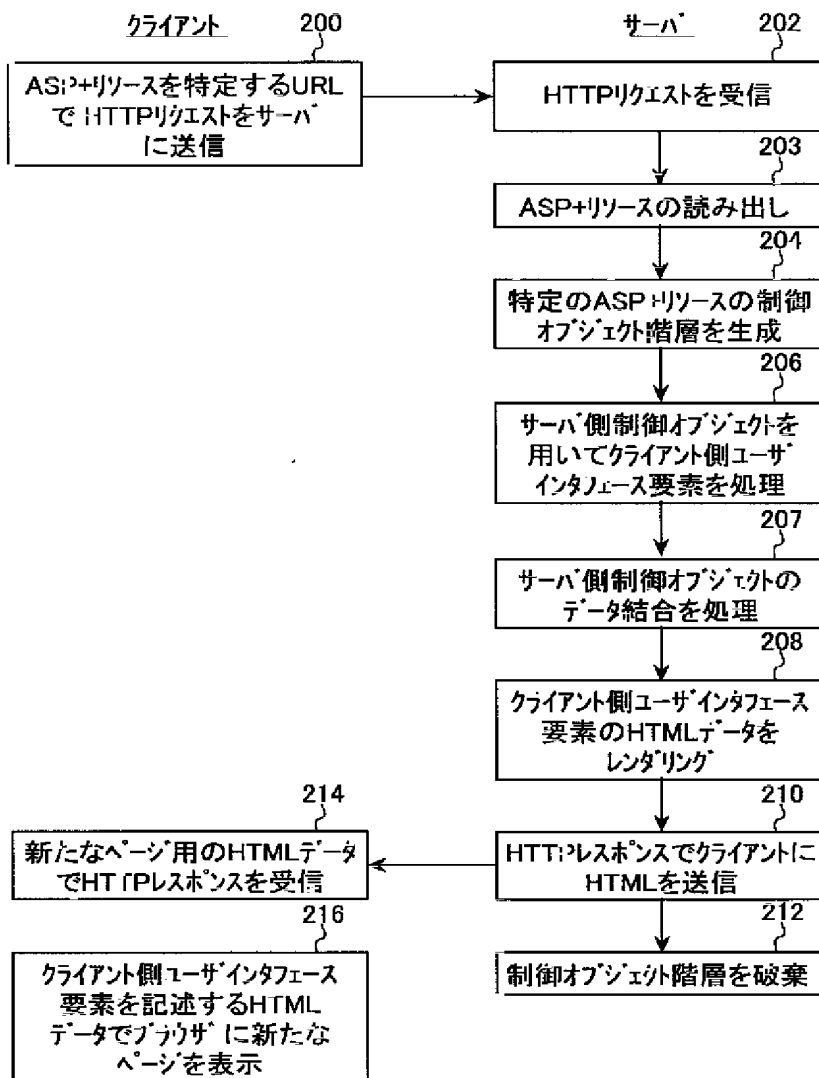
【図1】



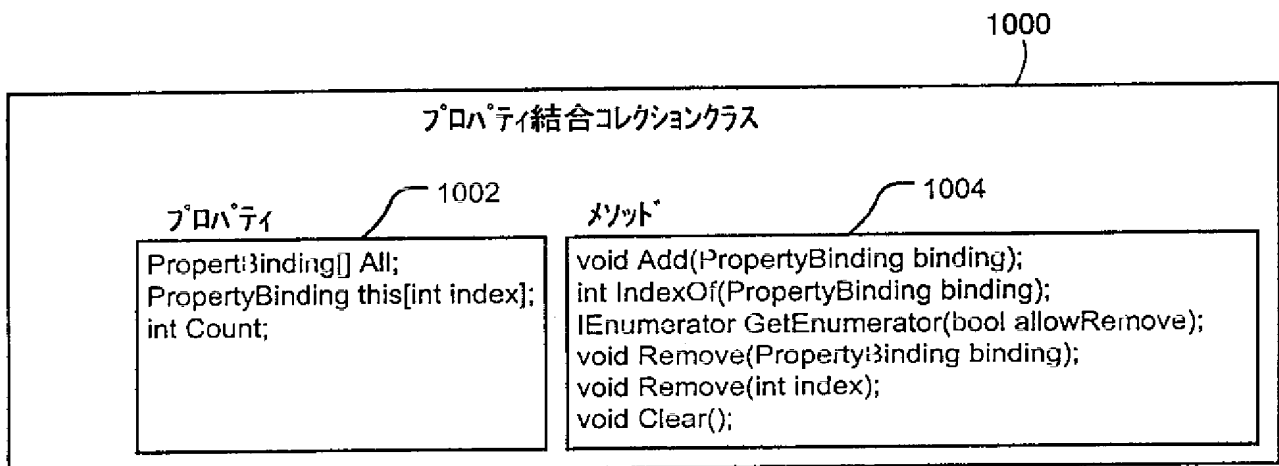
【図5】



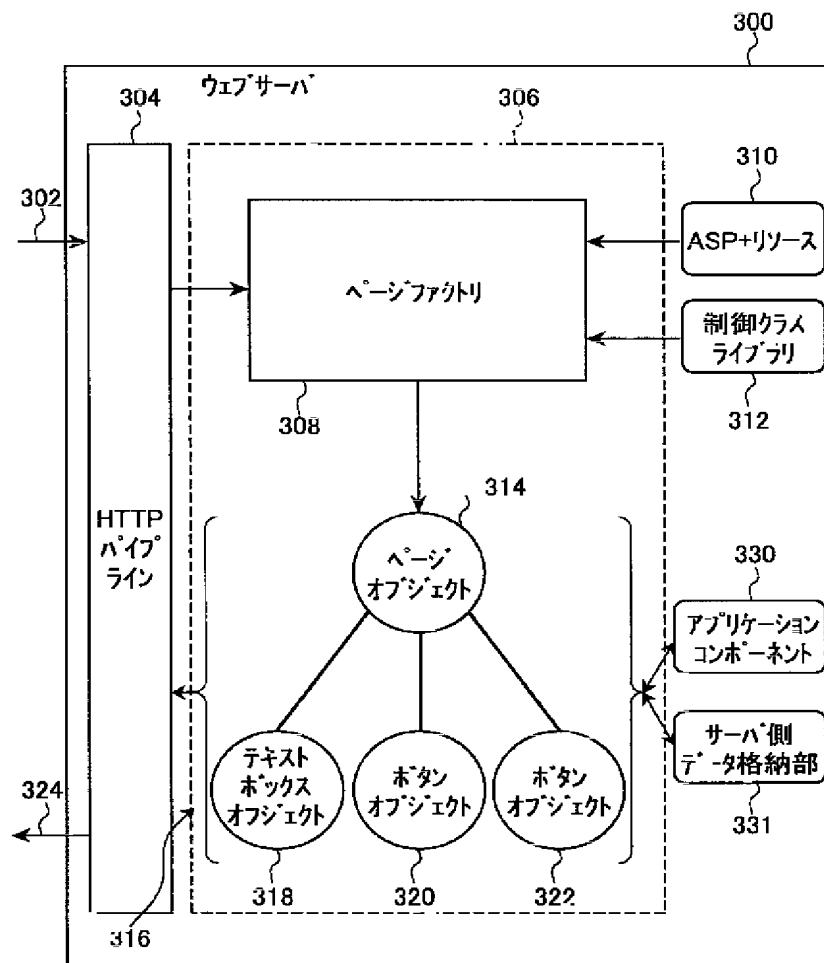
【図2】



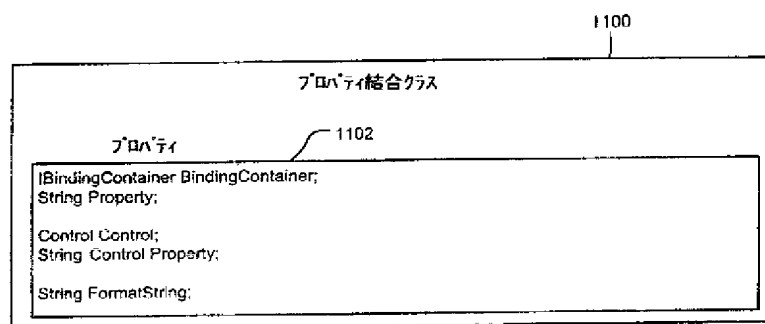
【図10】



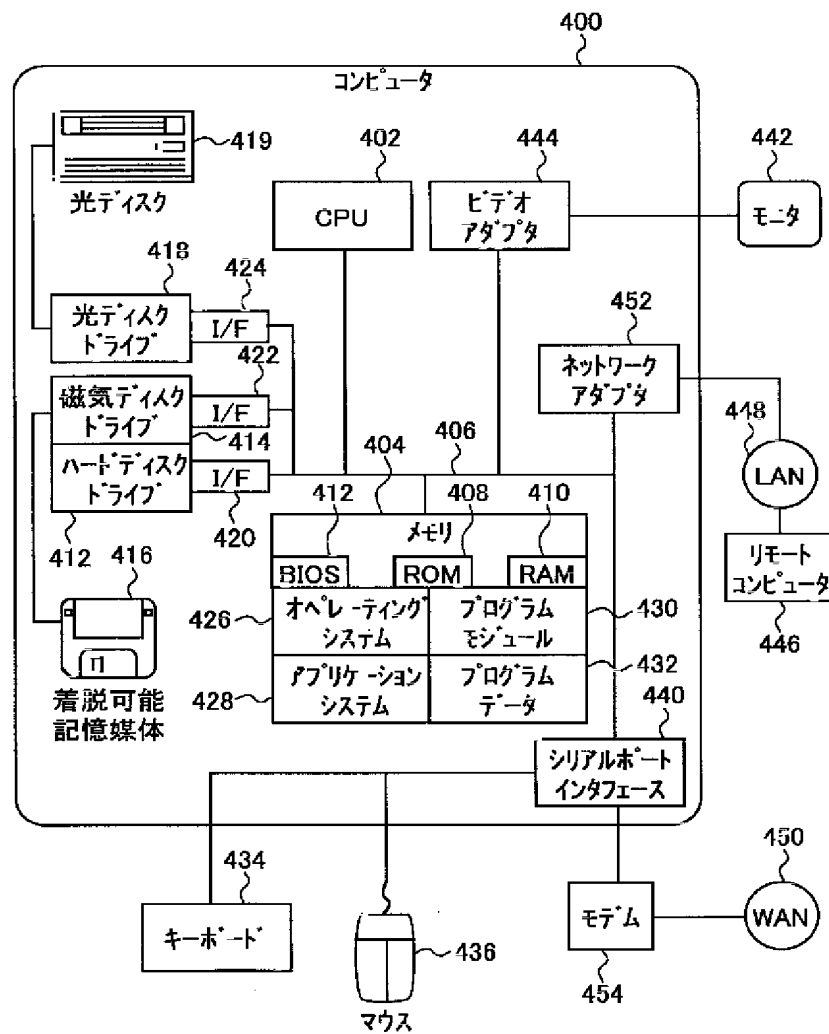
【図3】



【図11】



【図4】



【図6A】

600

```

1 <html>
2   <script runat=server>
3     Public MyOrderSystem as New OrderSystem
4     Public MyOrders() as Order
5     Public MyIngredients() as Ingredients
6
7     Overrides Sub Load()
8       Set MyOrders = MyOrderSystem.GetOrdersFromDay(Now)
9       Set MyItems = MyOrderSystem.GetItemFromOrders(MyOrders.DataItem.OrderID)
10    End Sub
11  </script>
12  <body>
13    <H1> Here are today's orders: </H1>
14
15    <form runat=server>
16      <wfc:tablelist databinding="datasource:Page.MyOrders" runat=server>
17        <template:itemtemplate>
18          <span databinding="Text:DataItem.OrderID" runat=server/>
19          <span databinding="Text:DataItem.Quantity" runat=server/>
20          <wfc:tablelist databinding="datasource:MyOrders.MyIngredients" runat=server>
21            <template:itemtemplate>
22              <span databinding="Text:Index" runat=server/>
23              <span databinding="Text:DataItem.Item" runat=server/>
24            </template:itemtemplate>
25          </wfc:tablelist>
26        </template:itemtemplate>
27      </wfc:tablelist>
28    </form>
29  </body>
30 </html>

```



【図6B】

602

```

1  <html>
2  <script runat=server>
3      Public MyOrderSystem as New OrderSystem
4      Public MyOrders() as Order
5      Public MyIngredients() as Ingredients

6      Overrides Sub Save()
7          MyOrderSystem.UpdateOrders(MyOrders)
8          MyOrderSystem.UpdateItems(MyItems)
9      End Sub

10 </script>

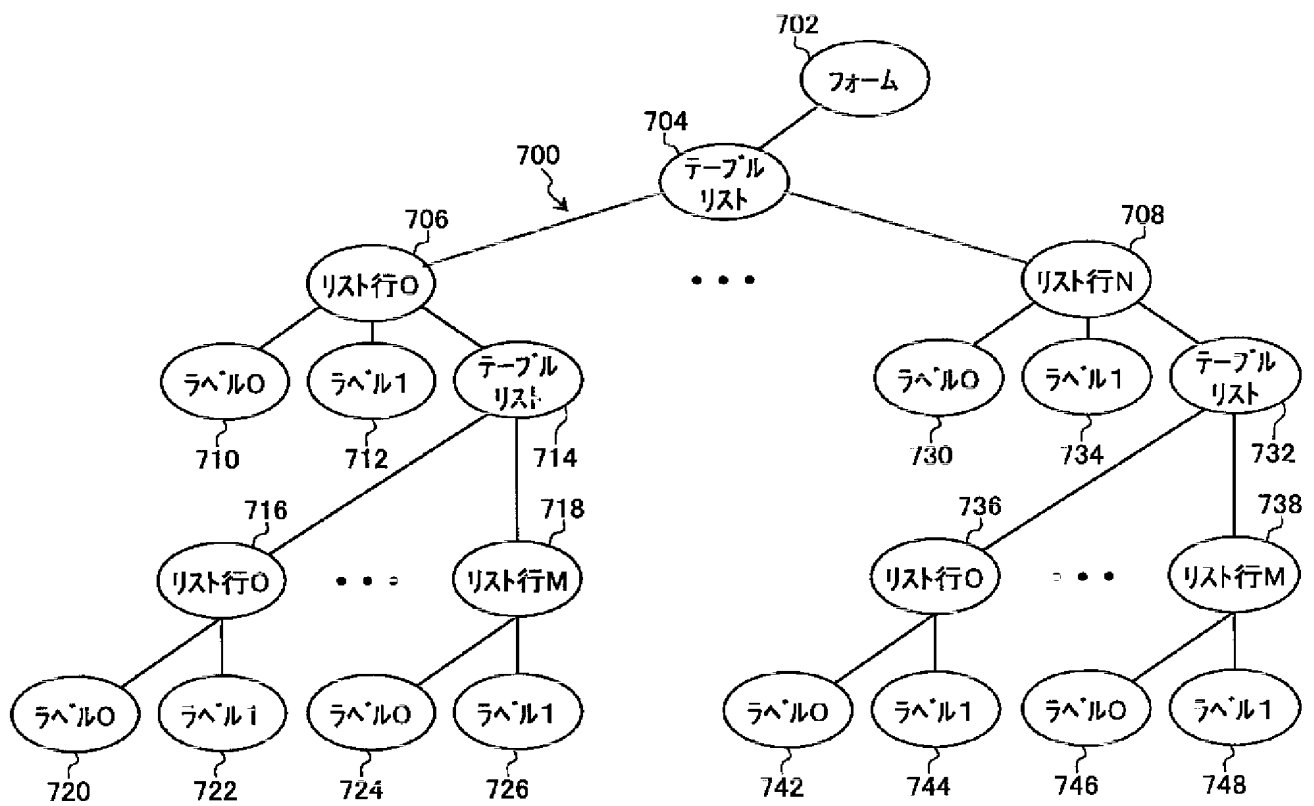
11 <body>
12     <H1> Here are today's orders: </H1>

13     <form runat=server>

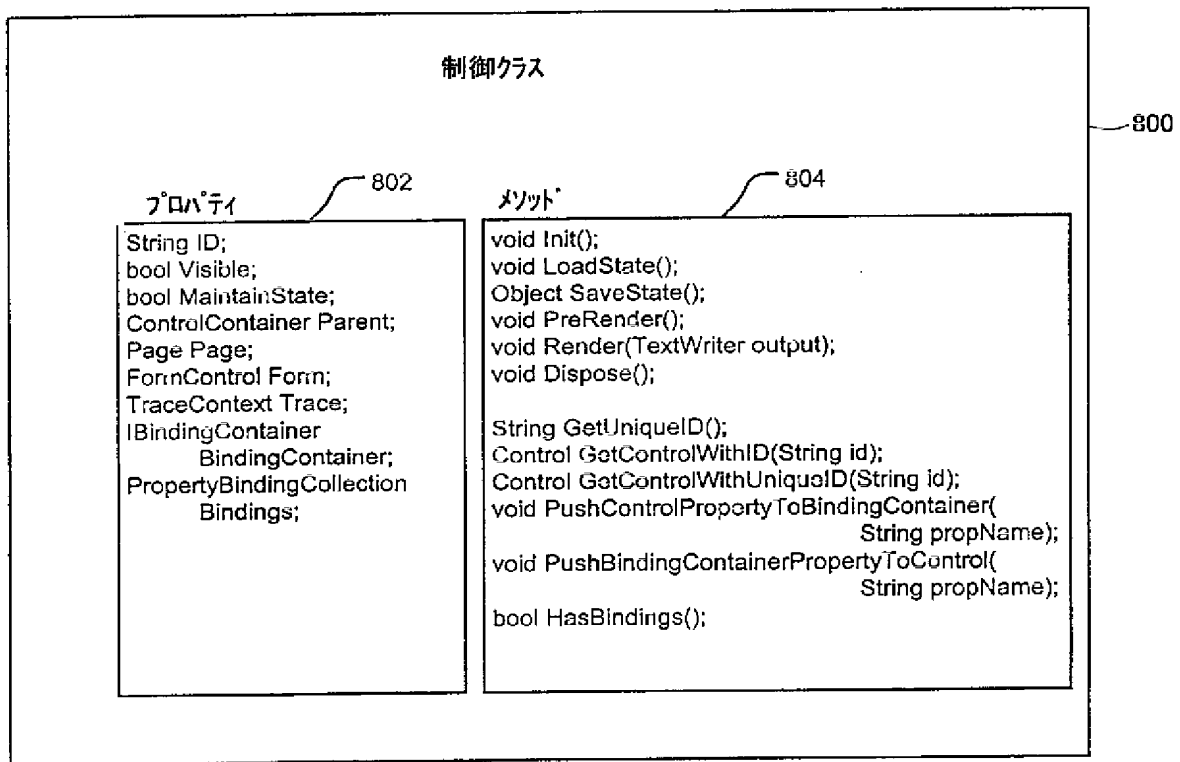
14         <wfc:tablelist databinding="datasource:Page.MyOrders" runat=server>
15             <template:itemtemplate>
16                 <span databinding="Text:DataItem.OrderID" runat=server/>
17                 <span databinding="Text:DataItem.Quantity" runat=server/>
18                 <wfc:tablelist databinding="datasource:MyOrders.MyIngredients" runat=server>
19                     <template:itemtemplate>
20                         <span databinding="Text:Index" runat=server/>
21                         <span databinding="Text:DataItem.Item" runat=server/>
22                     </template:itemtemplate>
23                 </wfc:tablelist>
24             </template:itemtemplate>
25         </wfc:tablelist>
26     </form>
27 </body>
28 </html>

```

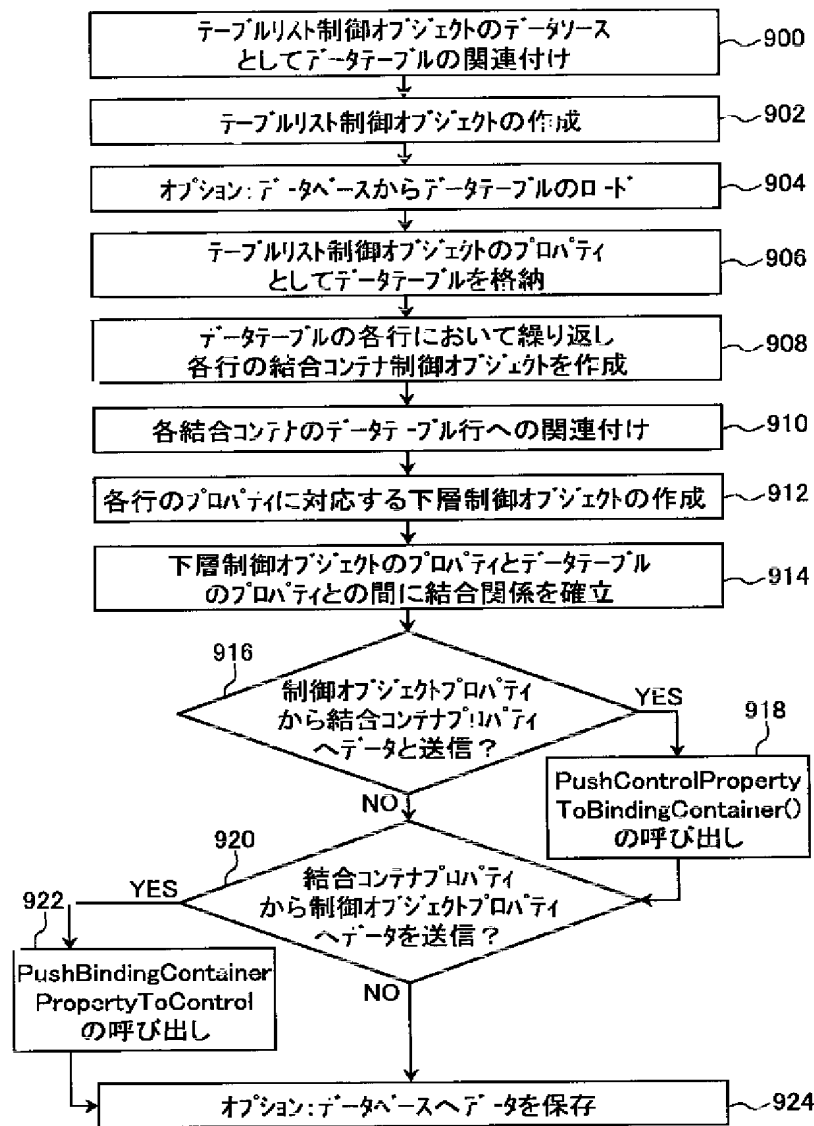
【図7】



【図8】



【図9】



## 【手続補正書】

【提出日】平成13年6月12日(2001.6.12)

## 【手続補正1】

【補正対象書類名】明細書

【補正対象項目名】0096

【補正方法】変更

【補正内容】

【0096】

【発明の効果】以上のように本発明によれば、1つ以上のサーバ側制御オブジェクトと1つ以上のサーバ側データ格納部とを結合する階層データを提供し、サーバ側制御オブジェクトのプロパティが、その結合コンテナを介

して、サーバ側データ格納部のプロパティに結合されるようにすることで、ユーザインタフェース要素を処理することが要求されるプログラミングを適切にカプセル化することができ、ページ開発のみならずアプリケーションプログラム開発についても、データベース構造やプログラミングに関する深い知識を必要とすることなく、開発労力の軽減を図ることが可能となる。

## 【手続補正2】

【補正対象書類名】明細書

【補正対象項目名】符号の説明

【補正方法】変更

【補正内容】

## 【符号の説明】

|         |                   |               |                  |
|---------|-------------------|---------------|------------------|
| 100     | クライアント            | 412           | ハードディスクドライブ      |
| 102     | ブラウザ              | 414           | 磁気ディスクドライブ       |
| 104     | ウェブページ            | 416           | 着脱可能記憶媒体         |
| 112     | HTTPレスポンス         | 418           | 光ディスクドライブ        |
| 114     | HTTPリクエスト         | 419           | 光ディスク            |
| 116     | ウェブサーバ            | 420、422、424   | I/F              |
| 118     | HTTPパイプライン        | 426           | オペレーティングシステム     |
| 120     | ハンドラ              | 428           | アプリケーションプログラム    |
| 122     | 静的コンテンツ資源         | 430           | プログラムモジュール       |
| 124     | 動的コンテンツ資源         | 432           | プログラムデータ         |
| 126     | サーバ側制御クラスライブラリ    | 434           | キーボード            |
| 128     | クライアント側制御クラスライブラリ | 436           | マウス              |
| 130     | 非UIサーバコンポーネント     | 440           | シリアルポートインタフェース   |
| 131     | サーバ側データ格納部        | 442           | モニタ              |
| 300     | ウェブサーバ            | 444           | ビデオアダプタ          |
| 304     | HTTPパイプライン        | 446           | リモートコンピュータ       |
| 308     | ページファクトリ          | 448           | LAN              |
| 310     | ASP+リソース          | 450           | WAN              |
| 312     | 制御クラスライブラリ        | 452           | ネットワークアダプタ       |
| 314     | ページオブジェクト         | 454           | モデム              |
| 318     | テキストボックスオブジェクト    | 702           | フォーム             |
| 320、322 | ボタンオブジェクト         | 704、714、732   | テーブルリスト          |
| 330     | アプリケーションコンポーネント   | 800           | 制御クラス            |
| 331     | サーバ側データ格納部        | 802、1002、1102 | プロパティ            |
| 400     | コンピュータ            | 804、1004      | メソッド             |
| 402     | CPU               | 1000          | プロパティ結合コレクションクラス |
| 404     | メモリ               | 1100          | プロパティ結合クラス       |

## フロントページの続き

(72)発明者 クーパー、ケネス ビー、  
アメリカ合衆国、98199 ワシントン州  
シアトル、ウエスト ブレイン 3410

(72)発明者 ギャスリー、スコット ディー、  
アメリカ合衆国、98052 ワシントン州  
レッドモンド、アパートメント ユー  
2102、エヌイー 90 ストリート  
17786

(72)発明者 エボ、デイビッド エス、  
アメリカ合衆国、98052 ワシントン州  
レッドモンド、アパートメント ジェイ  
139、アボンデイル ロード 10909

(72)発明者 アンダース、マーク ティー、  
アメリカ合衆国、98007 ワシントン州  
ベルビュー、エヌイー 12 ス プレイス  
14425

(72)発明者 ビータース、テッド エー、  
アメリカ合衆国、98119 ワシントン州  
シアトル、8 ス アベニュー ウェスト  
2431

(72)発明者 ミレット、スティーブン ジェイ、  
アメリカ合衆国、98026 ワシントン州  
エドモンズ、オリンピック ビュー ドラ  
イブ 8412

Fターム(参考) 5B082 GA07

【 外国語明細書 】

1 Title of Invention

DATABINDING USING SERVER-SIDE CONTROL OBJECTS

2 Claims

1. In a server computer coupled to a client computer system, a method of data binding a property of a descendent server-side control object to a property of a data set in a server-side data table having one or more data sets, wherein the descendent server-side control object corresponds to a client-side user interface element, the method comprising:

iterating over each data set of the server-side data table to create a binding container server-side control object corresponding to each data set;

associating each binding container server-side control object with one of the data sets of the server-side data table;

creating a descendent server-side control object for a property of each data set, a given descendent server-side control object being a descendent of the binding container server-side control object corresponding to a given data set and being associated with a property of the given data set; and

establishing a data binding relationship between the property of the descendent server-side control object and the property of one of the data sets of the server-side data table.

2. The method of claim 1 further comprising:

loading the server-side data table from a server-side database;

associating an iterating server-side control object with the server-side data table;

creating the iterating server-side control object in a server-side control object hierarchy, wherein each binding container server-side control object is a child of the iterating server-side control object; and

storing the server-side data table as a property of the iterating server-side control object.

3. The method of claim 2 wherein the operation of associating an iterating server-side control object comprises:

setting a data source property of the iterating server-side control object to reference the server-side data table.

4. The method of claim 2 wherein the iterating operation comprises:

creating each binding container server-side control element as a child of the iterating server-side control object, each binding container server-side control object being associated with a given data set of the server-side data table.

5. The method of claim 1 wherein the establishing operation comprises:

creating a binding object including a control identifier of the descendent server-side control object, a first property identifier of the property of the descendent server-side control object, a binding container identifier of the binding container server-side control object, a second property identifier of the property of the data set of the server-side data table.

6. The method of claim 1 further comprising:

transmitting data stored as the property of the descendent control object for storage as the property of the data set of the server-side data table, based on the data binding relationship.

7. The method of claim 1 further comprising:

receiving data stored as the property of the of the server-side data table for storage in the property of the descendent control object, based on the data binding relationship.

8. In a server computer coupled to a client computer system, a computer data signal embodied in a carrier wave by a computing system and encoding a computer program for executing a computer process data binding a property of a descendent server-side control object to a property of a data set in a server-side data table having one or more data sets, wherein the descendent server-side control object corresponds to a client-side user interface element, the computer process comprising:

iterating over each data set of the server-side data table to create a binding container server-side control object corresponding to each data set;

associating each binding container server-side control object with one of the data sets of the server-side data table;

creating a descendent server-side control object for a property of each data

set, a given descendent server-side control object being a descendent of the binding container server-side control object corresponding to a given data set and being associated with a property of the given data set; and

establishing a data binding relationship between the property of the descendent server-side control object and the property of one of the data sets of the server-side data table.

9. In a server computer coupled to a client computer system, a computer program storage medium readable by a computer system and encoding a computer program for executing a computer process data binding a property of a descendent server-side control object to a property of a data set in a server-side data table having one or more data sets, wherein the descendent server-side control object corresponds to a client-side user interface element, the computer process comprising:

iterating over each data set of the server-side data table to create a binding container server-side control object corresponding to each data set;

associating each binding container server-side control object with one of the data sets of the server-side data table;

creating a descendent server-side control object for a property of each data set, a given descendent server-side control object being a descendent of the binding container server-side control object corresponding to a given data set and being associated with a property of the given data set; and

establishing a data binding relationship between the property of the descendent server-side control object and the property of one of the data sets of the server-side data table.

10. A computer program product encoding a computer program for executing on a computer system a computer process for data binding a property of a descendent server-side control object to a property of a server-side data array having one or more data objects, wherein the child server-side control object corresponds to a client-side user interface element, the computer process comprising:

loading the server-side data array from a server-side database;

associating an iterating server-side control object with the server-side data table;



creating the iterating server-side control object in a server-side control object hierarchy;

storing the server-side data array as a property of the iterating server-side control object;

create a binding container server-side control object corresponding to one of the data objects, wherein the binding container server-side control object is a child of the iterating server-side control object;

associating the binding container server-side control object with the one of the data objects of the server-side data array;

creating a descendent server-side control object for each property of each data object each descendent server-side control object being a descendent of the binding container server-side control object; and

establishing a data binding relationship between the property of the descendent server-side control object and the property of the server-side data array.

11. The computer program product of claim 10 wherein the operation of associating an iterating server-side control object comprises:

setting a data source property of the iterating server-side control object to reference the server-side data array.

12. The computer program product of claim 11 wherein the iterating operation of associating the iterating server-side control object comprises:

creating each binding container server-side control element as a child of the iterating server-side control object, each binding container server-side control object being associated with a given data object of the server-side data array.

13. The computer program product of claim 10 wherein the establishing operation comprises:

creating a binding relationship object including a control identifier of the descendent server-side control object, a property identifier of the property of the descendent server-side control object, a binding container identifier of the binding container server-side control object, an property identifier of the property of the data object of the server-side data array.

14. The computer program product of claim 10 further comprising:  
sending data stored as the property of the descendent control object for storage as the property of the data object of the server-side data array, based on the data binding relationship.

15. The computer program product of claim 10 further comprising:  
receiving data stored as the property of the data object of the server-side data array for storage in the property of the descendent control object, based on the data binding relationship.

16. A server for performing server-side data binding using a hierarchy of server-side control objects corresponding to client-side user interface elements, the server comprising:

- a server-side data array having one or more data objects, each data object including a property;
- an iterating server-side control object in a server-side control object hierarchy and being associated with the server-side data array;
- one or more binding container server-side control objects iteratively created by the iterating server-side control object based on a number of data objects in the server-side data array;
- one or more descendent server-side control objects corresponding to a property of each data object of the server-side data array, each descendent server-side control being created as a child of the binding control server-side control object of a given data object in the server-side data array; and
- a data binding relationship structure describing a data binding relationship between a property of the descendent control object and the property of the data object of the server-side data array.

17. The server of claim 16 further comprising:  
a push module storing data from the property of the descendent control object to the property of the data object of the server side data array, based on the data binding relationship.

18. The server of claim 16 further comprising:  
a push module storing data from the property of the data object of the server side data array to the property of the descendent control object, based on the data binding relationship.

19. The server of claim 16 further comprising:  
a portion of a server-side datastore loaded into the server-side data array.

20. The server of claim 16 further comprising:  
a portion of a server-side datastore into which the server-side data array is saved.

**Technical Field**

The invention relates generally to a web server framework, and more particularly to server-side control objects that bidirectionally bind with properties of server-side data tables.

**Background of the Invention**

A typical web browser receives data from a web server that defines the appearance and rudimentary behavior of a web page for display on a client system. In a typical scenario, a user specifies a Uniform Resource Locator ("URL"), a global address of a resource on the World Wide Web, to access a desired web site. Generally, the term "resource" refers to data or routines that can be accessed by a program. An example URL is "http://www.microsoft.com/ms.htm". The first part of the example URL indicates a given protocol (i.e., "http") to be used in the communication. The second part specifies the domain name (i.e., "www.microsoft.com") where the resource is located. The third part specifies the resource (i.e., a file called "ms.htm") within the domain. Accordingly, a browser generates an HTTP (HyperText Transport Protocol) request associated with the example URL to retrieve the data associated with ms.htm file within the www.microsoft.com domain. A web server hosting the www.microsoft.com site receives the HTTP request and returns the requested web page or resource in an HTTP response to the client system for display in the browser.

The "ms.htm" file of the example above includes static HTML (HyperText Markup Language) code. HTML is a plain-text authoring language used to create documents (e.g., web pages) on the World Wide Web. As such, an HTML file can be retrieved from a web server and displayed as a web page in a browser to present the rich graphical experience that users have come to expect while viewing information from the Internet. Using HTML, a developer can, for example, specify formatted text, lists, forms, tables, hypertext links, inline images and sounds, and

background graphics for display in the browser. An HTML file, however, is a static file that does not inherently support dynamic generation of web page content.

If dynamic content is to be displayed, such as a changing stock price or traffic information, a server-side application program is generally developed to handle the more complex client-server interaction. The server-side application program processes an HTTP request and generates the appropriate HTML code for transmission to the client in an HTTP response. An exemplary HTTP request may include parameters, such as data in a query string or data from web-based forms. As such, a server-side application program can process these parameters and dynamically generate HTML code for transmission in an HTTP response to the client. An exemplary server-side application program may generate documents containing appropriate HTML code using a sequence of one or more formatted text write operations to a memory structure. Thereafter, the resulting document is transmitted to a client system in an HTTP response, where it is displayed as a web page in the browser.

Developing a server-side application program can be a complex task requiring not only familiarity with normal HTML coding that is used to layout a Web page, but also with programming basics, including one or more programming languages (e.g., C++, Perl, Visual Basic, or Jscript). Web page designers, on the other hand, are frequently graphics designers and editors, who may lack programming experience. Furthermore, simplifying complex web page development can speed the development of new web content by any developer. Generally, development of a custom server-side application program also requires tremendous effort, so much, in fact, that developers are often disinclined to attempt it. It is desirable, therefore, to provide a development framework that allows a developer to dynamically create and process a web page with minimal programming.

One approach to minimize the programming requirements of dynamic web page generation has been the Active Server Page (ASP) framework, provided by Microsoft Corporation. An ASP resource typically includes Visual Basic or Jscript code, for example, to process an HTTP request that specifies the ASP resource as the desired resource and, thereafter, to generate the resulting HTML code in a HTTP response to the client. Furthermore, an ASP resource may reference pre-developed

or third party client-side library components (e.g., client-side ACTIVEX controls) to ease a given application programming effort. However, in the current server-side application frameworks, the programming required to dynamically manage client-side user interface elements (e.g., text boxes, list boxes, buttons, hypertext links, images, sounds, etc.) within server-side applications can still require sophisticated programming skills and considerable effort. An unanswered problem exists in properly encapsulating programming required to process user interface elements, so as to allow the web page developer to focus on other aspects of the web page.

Certain client-side user interface elements include data associated with a server-side datastore, such as a database. An example of such an element may be a client-side table displaying a selection of products and prices from a server-side product database. Another example may include a text box in which a consumer can enter a shipping address to be stored to a server-side customer database for later use. Traditionally, the client-side user-interface elements are written into the page by a page developer, and an application program is written by an application program developer to perform the data communications between the page and the server-side databases. A problem is that both page developing and application program developing require significant development effort and intimate knowledge of the database structures and some level of complicated programming (e.g., through client-side scripting or server-side application programming).

#### **Summary of the Invention**

In accordance with the present invention, the above and other problems are solved by providing hierarchical data binding between one or more server-side control objects and one or more server-side datastores. Hierarchical data binding is supported by automatically creating a binding container for each data object in a data array. Each binding container provides its children with access to its public properties, including the data objects of the server-side datastore with which it is associated. As such, the property of a server-side control object may be bound through its binding container to a property of a server-side datastore. A data binding relationship may be established, without limitation, unidirectionally from a property of the server-side control object to the property of the server-side datastore or vice

versa. A data binding relation may also be established, without limitation, bidirectionally between a property of the server-side control object and the property of the server-side datastore.

In one implementation of the present invention, a method of data binding a property of a descendent server-side control object to a property of a data set (e.g., a table row) of a server-side data table having one or more data sets is provided in a server computer coupled to a client computer system. The descendent server-side control object corresponds to a client-side user interface element. An iterating control object iterates over each data set of the server-side data table to create a binding container server-side control object corresponding to each data set. Each binding container server-side control object is associated with a data set of the server-side data table. A descendent server-side control object is created for a property of each data set. A given descendent server-side control object is a descendent of the binding container server-side control object corresponding to a given data set and is associated with a property of the given data set. A data binding relationship is established between the property of the descendent server-side control object and the property of one of the data sets of the server-side data table.

In other implementations of the present invention, articles of manufacture are provided as computer program products. One embodiment of a computer program product provides a computer program storage medium readable by a computer system and encoding a computer program for executing a computer process data binding a property of a descendent server-side control object to a property of a server-side data array having one or more data objects is provided. Another embodiment of a computer program product may be provided in computer data signal embodied in a carrier wave by a computing system and encoding the computer program. The child server-side control object corresponds to a client-side user interface element. The server-side data array is loaded from a server-side database. An iterating server-side control object is associated with the server-side data table and is created into a server-side control object hierarchy. The server-side data array is stored as a property of the iterating server-side control object. A binding container server-side control object corresponding to one of the data objects is created as a child of the iterating server-side control object. The binding container server-side control object is associated with one of the data objects of the server-side

data array. A descendent server-side control object is created for each property of each data object. Each descendent server-side control object is a descendent of the binding container server-side control object. A data binding relationship is established between the property of the descendent server-side control object and the property of the server-side data array.

In another implementation, a server for performing server-side data binding using a hierarchy of server-side control objects corresponding to client-side user interface elements is also provided, including a server-side data array having one or more data objects. Each data object may include a property. An iterating server-side control object in a server-side control object hierarchy is associated with the server-side data array. One or more binding container server-side control objects are iteratively created by the iterating server-side control object based on a number of data objects in the server-side data array. One or more descendent server-side control objects corresponds to a property of each data object of the server-side data array. Each descendent server-side control is created as a descendent of the binding container server-side control object of a given data object in the server-side data array. A data binding relationship structure describes a data binding relationship between a property of the descendent control object and the property of the data object of the server-side data array.

In an embodiment of the present invention, a web page content is dynamically generated on a web server for display on a client. The client can be, for example, any browser that supports standard HTML or any other authoring language. The client in the web server communicates across the network, such as by HTTP requests and HTTP responses. As such, the web server generates web page content, possibly in the form of HTML code, and transmits the content to the client, which can display the web page in a browser. Server-side control objects, which may logically correspond to individual user interface elements of the web page, are created on the web server to process and render the web page content. The server-side control objects are declared in a dynamic content resource, such as an ASP+ resource, which is processed by a page factory to create a hierarchy of server-side control object. The control objects in the hierarchy cooperate to process the request

received from the client and then generate resulting web page content for transmission to the client, before the control objects in a hierarchy are terminated.

A page object may instantiate as the top level of the control object hierarchy in an embodiment of the present invention. A page object, which is also a control object, typically contains one or more child control objects, and each child control object can contain one or more child control object of its own to extend into the hierarchy of multiple levels. The page object and descendent control objects execute a sequence of operations to process and generate the web content that corresponds to client-side user interface elements.

One of the operations in the sequence includes a data binding operation that controls communications between one or more of the server-side control objects and a server-side datastore. In an embodiment present invention, a server-side datastore is a database storing data related to user interface elements of the web page (e.g., see datastore 131 in FIG. 1). Other non-database types of datastores are also contemplated within the scope of the present invention, including without limitation configuration registries, data files, data streams. By declaring server-side control objects and associated data binding relationships in a dynamic content resource, a page developer can configure a control object hierarchy having unidirectional and/or bidirectional data binding relationships with a server-side data array associated with the server-side datastore.

FIG. 1 illustrates a web server for dynamically generating web page content for display on a client in an embodiment of the present invention. A client 100 executes a browser 102 that displays a web page 104 on a display device of the client 100. The client 100 may include a client computer system having a display device, such as a video monitor. An "INTERNET EXPLORER" browser, marketed by Microsoft Corporation, is an example of a browser 102 in an embodiment of the present invention. Other exemplary browsers include without limitation "NETSCAPE NAVIGATOR" and "MOZILLA". The exemplary web page 104 incorporates a table list element 140 having four list row elements. The list row element 142 is a child user interface element contained by the table list element 140. The list row element 142 is defined by a template declaration (see FIG. 6A for an exemplary template declaration) and has two columns containing label elements 144 (e.g., representing an OrderID) and 146 (e.g., representing a quantity of the product



ordered), and one column containing a table list element 148. The elements 144, 146, and 148 are declared as child user interface elements contained by the list row element 142 (see FIG. 6A for exemplary label and table list declarations).

Additional list row elements 150, 152, and 154 are also shown in web page 104.

In accordance with the declarations of FIG. 6A, server-side control objects corresponding to the user interface elements of the web page 104 are data bound to two server-side data arrays (e.g., server-side data tables, MyOrders and MyItems, associated with server-side database access through the MyOrderSystem object). Generally, a server-side data array is associated with (e.g., loaded from and/or stored to) a server-side datastore and includes an index data set (e.g., data rows or data objects) within the control object hierarchy. Such data sets may have properties, which for example, may represent a data value of a data table row. The column 148 represents a sub-table list element having two columns: (1) a column of row indexes from a bound Items data table; (2) a column of three list row elements containing labels (e.g., item names) from the Items data table.

The browser 102 can receive HTML code in the HTTP response 112 from a web server 116 and displays the web page as described by the HTML code. Although HTML is described with reference to one embodiment, other authoring languages, including without limitation SGML (Standard Generalized Markup Language), XML (eXtensible Markup Language), and WML (Wireless Markup Language), which is an XML-based markup language, designed for specifying the content and user interfaces of narrowband wireless devices, such as pagers and cellular phones, are contemplated within the scope of the present invention. Furthermore, although standard HTML 3.2 is primarily disclosed herein, any version of HTML is supportable within the scope of the present invention.

The communications between the client 100 and the web server 116 may be conducted using a sequence of HTTP requests 114 and HTTP responses 112. Although HTTP is described with reference to one embodiment, other transport protocols, including without limitation S-HTTP, are contemplated within the scope of the present invention. On the web server 116, an HTTP pipeline module 118 receives an HTTP request 114, resolves the URL, and invokes an appropriate handler 120 for processing the request. In an embodiment of the present invention, a

plurality of handlers 120 to handle different types of resources are provided on the web server 116.

For example, if the URL specifies a static content resource 122, such as an HTML file, a handler 120 accesses the static content resource 122 and passes the static content resource 122 back through the HTTP pipeline 118 for communication to the client 100 in an HTTP response 112. Alternatively, in an embodiment of the present invention, if the URL specifies a dynamic content resource 124, such as an ASP+ resource, a handler 120 accesses the dynamic content resource 124, processes the contents of the dynamic content resource 124, and generates the resulting HTML code for the web page 104. In an embodiment of the present invention, the resulting HTML code includes standard HTML 3.2 code. Generally, a dynamic content resource is a server-side declaration datastore that can be used to dynamically generate the authoring language code that describes a web page to be displayed on a client. The HTML code for the web page is then passed through the HTTP pipeline 118 for communication to the client 100 in an HTTP response 112.

During its processing, a handler 120 can also access libraries of pre-developed or third party code to simplify the development effort. One such library is a server-side class control library 126, from which the handler 120 can instantiate server-side control objects for processing user interface elements and generating the resultant HTML data for display of a web page. In an embodiment of the present invention, one or more server-side control objects map to one or more user interface elements, visible or hidden, on the web page described in the dynamic content resource 124.

A second library, in contrast, is a client-side control class library 128, such as a library including "ACTIVEX" components from Microsoft Corporation. An "ACTIVEX" control is a COM ("Component Object Model") object that follows certain standards in how it interacts with its client and other components. A client-side "ACTIVEX" control, for example, is a COM-based component that can be automatically downloaded to a client and executed by a web browser on the client. Server-side ACTIVEX components (not shown) are COM-based components that may be implemented on a server to perform a variety of server-side functions, such as providing the server-side functionality of a stock price look-up application or

database component. A more detailed discussion of ACTIVEX can be found in "Understanding ACTIVEX and OLE", David Chappell, Microsoft Press, 1996.

In contrast to "ACTIVEX" controls, a server-side control object in an embodiment of the present invention, being specified in a dynamic content resource 124, logically corresponds to a user interface element that is incorporated in the web page on the client. The server-side control object can also generate valid HTML code that can include, for example, an HTML tag and a locator referencing a given client-side "ACTIVEX" control. If the browser already has the code for the client-side "ACTIVEX" control within its storage system, the browser executes the "ACTIVEX" control within the web page on the client. Otherwise, the browser downloads the code for the "ACTIVEX" control from the resource specified by the locator and then executes the "ACTIVEX" control within the web page on the client. A server-side control object in an embodiment of the present invention can also raise events to a server-side "ACTIVEX" control used to implement a stock look-up application on the server.

A handler 120 also has access to one or more non-user-interface server components 130 that execute on the web server 116 or on another accessible web server. A non-user-interface server component 130, such as a stock price look-up application or database component, may be referenced in or associated with a dynamic content resource 124 that is processed by a handler 120. Server-side events raised by the control objects declared in the dynamic content resource 124 may be processed by server-side code, which calls appropriate methods in the non-user-interface component 130. As a result, the processing provided by the server-side control objects simplifies the programming of the non-user-interface server component 130 by encapsulating the processing and generation of the user interface elements of a web page, which allows the developer of the non-user-interface server component 130 to concentrate on the specific functionality of the application, rather than on user interface issues.

In an embodiment of the present invention, properties of server-side control objects are data bound to properties in a server-side datastore 131. Data binding allows a page developer to associate properties of control objects in the server-side control object hierarchy to data items in a server-side database, for example. During processing, each control object updates its data bound properties by retrieving an

associated property value from a data object in the database (referred to as unidirectional data binding from the server-side datastore). Alternatively, each control object updates a property of a server-side datastore using the control object's internal property value (referred to as unidirectional data binding to the server-side data store). The two types of unidirectional data binding can be combined to provide bidirectional data binding between server-side control objects and server-side datastores.

FIG. 2 illustrates a flow diagram of operations for processing and generating client-side user interface elements using server-side control objects in an embodiment of the present invention. In transmitting operation 200, the client transmits an HTTP request to the server. The HTTP request includes a URL that specifies a resource, such as an ASP+ resource. In receiving operation 202, the server receives the HTTP request and invokes the appropriate handler for processing the specified resource. The ASP+ resource is read in parsing operation 203. Creation operation 204 generates a server-side control object hierarchy based on the contents of the specified dynamic content resource (e.g., the ASP+ resource).

In operation 206, the server-side control objects of the control object hierarchy perform one or more of the following operations: postback event handling, postback data handling, state management, and data binding. Postback events and data (collectively "postback input") from user interface elements are communicated from the client to the server for processing. A postback event, for example, may include without limitation a "mouse click" event from a client-side button element or a "data change" event from a client-side textbox element that is communicated to the server. Postback data, for example, may include without limitation text entered by a user in a text box element or an index of an item selected from a drop-down box.

In operation 208, each server-side control object in the hierarchy is called to generate (or render) data, such as HTML code, for display of client-side user interface elements in the web page. Note that, although the term "render" may be used to describe the operation of displaying graphics on a user interface, the term "render" is also used herein to describe the operation of generating authoring language data that can be interpreted by client-application, such as a browser, for display and client-side functionality. A more detailed discussion of the processing

operation 206 and the rendering operation 208 is provided in association with FIG. 6. In one embodiment, calls to render() methods in individual control objects are performed using a tree traversal sequence. That is, a call to the render() method of a page object results in recursive traversal throughout appropriate server-side control objects in the hierarchy. Alternative methods for calling the render() methods for appropriate control objects may also be employed, including an event signaling or object registration approach. The parentheses designate the "render()" label as indicating a method, as compared to a data value.

In an embodiment of the present invention, the actual creation of the individual server-side control objects may be deferred until the server-side control object is accessed (such as when handling postback input, loading a state, rendering HTML code from the control object, etc.) in operations 206 or 208. If a server-side control object is never accessed for a given request, deferred control object creation optimizes server processing by eliminating an unnecessary object creation operation.

Transmitting operation 210 transmits the HTML code to the client in an HTTP response. In receiving operation 214, the client receives the HTML code associated with a new web page to be displayed. In display operation 216, the client system incorporates (e.g., displays) the user interface elements of the new page in accordance with the HTML code received from the HTTP response. It should be understood, however, that incorporation of a user-interface element may include non-display operations, such as providing audio or tactile output, reading and writing to memory, controlling the operation of scripts, etc. In termination operation 212, the server-side control object hierarchy is terminated. In an embodiment of the present invention, server-side control objects in the hierarchy are created in response to an HTTP request referencing an associated ASP+ resource, and destroyed subsequent to the rendering of authoring language data (e.g., HTML data). In an alternative embodiment, operation 212 may be performed after operation 208 and before operation 210.

FIG. 3 illustrates exemplary modules in a web server used in an embodiment of the present invention. The web server 300 receives an HTTP request 302 into the HTTP pipeline 304. The HTTP pipeline 304 may include various modules, such as modules for logging of web page statistics, user authentication, user authorization, and output caching of web pages. Each incoming HTTP request 302 received by the

web server 300 is ultimately processed by a specific instance of an HTTPHandler class (shown as handler 306). The handler 306 resolves the URL request and invokes an appropriate handler factory (e.g., a page factory module 308).

In FIG. 3, a page factory module 308 associated with the ASP+ resource 310 is invoked to handle the instantiation and configuration of the ASP+ resource 310. In one embodiment, an ASP+ resource can be identified by designating a particular suffix (or a file extension such as ".aspx") with the resource. When a request for a given ASP+ resource is first received by the page factory module 308, the page factory module 308 searches the file system for the appropriate file (e.g., the .aspx file 310). The file may contain text (e.g., authoring language data) or another data format (e.g., byte-code data or encoded data) that may later be interpreted or accessed by the server to service the request. If the physical file exists, the page factory module 308 opens the file and reads the file into memory. If the file cannot be found, the page factory module 308 returns an appropriate "file not found" error message.

After reading the ASP+ resource 310 into memory, the page factory module 308 processes the file content to build a data model of the page (e.g., lists of script blocks, directives, static text regions, hierarchical server-side control objects, server-side control properties, etc.). The data model is used to generate a source listing of a new object class, such as a COM+ ("Component Object Model+") class, that extends the page base class. The page base class includes code that defines the structure, properties, and functionality of a basic page object. In an embodiment of the present invention, the source listing is then dynamically compiled into an intermediate language and later Just-In-Time compiled into platform native instructions (e.g., X86, Alpha, etc.). An intermediate language may include general or custom-built language code, such as COM+ IL code, Java bytecodes, Modula 3 code, SmallTalk code, and Visual Basic code. In an alternative embodiment, the intermediate language operations may be omitted, so that the native instructions are generated directly from the source listing or the source file (e.g., the ASP+ resource 310). A control class library 312 may be accessed by the page factory module 308 to obtain predefined server-side control classes used in the generation of the control object hierarchy.

The page factory module 308 outputs a page object 314, which is a server-side control object that corresponds to the web page 104 of FIG. 1. The page object 314 and its children comprise an exemplary control object hierarchy 316. Other exemplary control objects are also contemplated in accordance with the present invention, including without limitation objects corresponding to the HTML controls in Table 1, as well as custom control objects. The page object 314 logically corresponds to the web page 104 of FIG. 1 and is hierarchically related to other control objects on the server. In one embodiment, a page object is a container object that hierarchically contains its children control objects. In an alternative embodiment, other forms of hierarchical relation may be employed, including a dependency relationship. In a more complex control object hierarchy with multiple levels of children, a child object can be a container object for other child objects (e.g., the table list 140 is a container for the list row elements 142, 150, 152, and 154 and their children). These server-side control objects cooperate to handle input from the HTTP request 302, to manage the states of server-side control objects, to perform data binding, and to render authoring language data (e.g., HTML) for a resulting web page for the client. The resulting HTML code is output from the control object hierarchy 316 and transmitted to the client in an HTTP response 324.

In the illustrated embodiment, the control objects in the control object hierarchy 316 are created and executed on the server 300, and each server-side control object logically corresponds to a user interface element on the client. The server-side control objects also cooperate to handle postback input from the HTTP request 302, to manage the states of server-side control objects, to perform data binding with server-side databases, and to generate authoring language data (e.g., HTML code) used to display a resulting web page at the client. The resulting authoring language data is generated (i.e., rendered) from the server-side control object hierarchy 316 and transmitted to the client in an HTTP response 324. For example, resulting HTML code may embody any valid HTML construct and may reference ACTIVEX-type control, JAVA applets, scripts, and any other web resources that yield client-side user interface elements (e.g., control buttons, text boxes, etc.) when processed by a browser.

By virtue of declarations made in the ASP+ resource 310, server-side control objects may also access one or more non-user-interface server components 330 to

provide interaction between the non-user-interface server component 330 and client-side user interface elements. For example, in response to postback input, server-side control objects can raise server-side events to the non-user-interface server components registered for those events. In this manner, the non-user-interface server component 330 can interact with the user through user interface elements without programming the code required to display and process these elements.

Properties of one or more server-side datastores 331 may be data bound to properties of the server-side control objects. A detailed discussion of an exemplary server-side control object hierarchy configured to support an embodiment of data binding is provided with regard to FIG. 7. The data binding relationships may be declared in the ASP+ resource and established during the compilation and runtime execution of the server-side control objects.

In summary, an embodiment of the present invention includes server-side control objects that are created and executed on the server to generate HTML code that is sent to a client. The HTML code may, for example, embody any valid HTML constructs and may reference ACTIVE-X-type controls, JAVA applets, scripts, and any other web resources to produce user interface buttons and other user interface elements at the client. A user at a client may interact with these user interface elements, which logically corresponds to the server-side control objects, and send a request back to the server. The server-side control objects are recreated on the server to process the data, events, and other characteristics of the user interface elements so as to generate the next round of HTML code to be transmitted in response to the client.

With reference to FIG. 4, an exemplary computing system for embodiments of the invention includes a general purpose computing device in the form of a conventional computer system 400, including a processor unit 402, a system memory 404, and a system bus 406 that couples various system components including the system memory 404 to the processor unit 400. The system bus 406 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 408 and random access memory (RAM) 410. A basic input/output system 412 (BIOS),



which contains basic routines that help transfer information between elements within the computer system 400, is stored in ROM 408.

The computer system 400 further includes a hard disk drive 412 for reading from and writing to a hard disk, a magnetic disk drive 414 for reading from or writing to a removable magnetic disk 416, and an optical disk drive 418 for reading from or writing to a removable optical disk 419 such as a CD ROM, DVD, or other optical media. The hard disk drive 412, magnetic disk drive 414, and optical disk drive 418 are connected to the system bus 406 by a hard disk drive interface 420, a magnetic disk drive interface 422, and an optical drive interface 424, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, programs, and other data for the computer system 400.

Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 416, and a removable optical disk 419, other types of computer-readable media capable of storing data can be used in the exemplary system. Examples of these other types of computer-readable mediums that can be used in the exemplary operating environment include magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), and read only memories (ROMs).

A number of program modules may be stored on the hard disk, magnetic disk 416, optical disk 419, ROM 408 or RAM 410, including an operating system 426, one or more application programs 428, other program modules 430, and program data 432. A user may enter commands and information into the computer system 400 through input devices such as a keyboard 434 and mouse 436 or other pointing device. Examples of other input devices may include a microphone, joystick, game pad, satellite dish, and scanner. These and other input devices are often connected to the processing unit 402 through a serial port interface 440 that is coupled to the system bus 406. Nevertheless, these input devices also may be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB). A monitor 442 or other type of display device is also connected to the system bus 406 via an interface, such as a video adapter 444. In addition to the monitor 442, computer systems typically include other peripheral output devices (not shown), such as speakers and printers.

The computer system 400 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 446. The remote computer 446 may be a computer system, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer system 400. The network connections include a local area network (LAN) 448 and a wide area network (WAN) 450. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, the computer system 400 is connected to the local network 448 through a network interface or adapter 452. When used in a WAN networking environment, the computer system 400 typically includes a modem 454 or other means for establishing communications over the wide area network 450, such as the Internet. The modem 454, which may be internal or external, is connected to the system bus 406 via the serial port interface 440. In a networked environment, program modules depicted relative to the computer system 400, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary, and other means of establishing a communication link between the computers may be used.

In an embodiment of the present invention, the computer 400 represents a web server, wherein the processor 402 executes a page factory module on an ASP+ resource stored on at least one of storage media 416, 412, 414, 418, 419, or memory 404. HTTP responses and requests are communicated over the LAN 448, which is coupled to a client computer 446.

FIG. 5 illustrates an operation flow diagram representing server-side processing of a page object and other control objects in an embodiment of the present invention. In operation 500, a page object constructor is called by the page factory module 308 (see FIG. 3). As a result, a page object (see e.g., the page object 314 in FIG. 3) is created to logically correspond to the web page user-interface element on the client. In operation 502, the page factory module calls the ProcessRequest member function of the page object, which initiates the staged operations for processing the HTTP request received from a client. In a first stage of one embodiment of the present invention, a server-side Create operation (not shown)

creates the descendant server-side control objects contained in the control object hierarchy of the page object, that is, constructors for child control objects are recursively called to create the control objects during the lifetime of the processing of the HTTP request.

In an alternate embodiment, however, creation of child control objects is deferred until the control object is required for a given processing step (e.g., handling a postback event, handling postback data, loading or saving a viewstate, resolving data binding, or rendering HTML code for the corresponding user interface element). The latter embodiment, which is said to implement "deferred control object creation", is an optimization that can alleviate unnecessary CPU and memory utilization. For example, a user input event received from the client may result in the creation of a completely different web page. In this case, it is unnecessary to instantiate an entire control object hierarchy of the previous page only to process an event that immediately results in the termination of the control object hierarchy and the instantiation of a new and different control object hierarchy for a new page.

In response to the server call to the page object's ProcessRequest method, operations 504 through 520 may be executed by the page object and by individual descendant control objects, depending in part on the data of a given HTTP request. In an embodiment of the present invention, the operations 504-520 are performed for each individual object in the order illustrated in FIG. 5; however, a given operation for one object may occur out of order or not at all with respect to a given operation of another object, depending on the HTTP request. For example, a first object may perform its Init operation 504 and its Load operation 506, and begin postback data processing operation 508, before a descendant control object performs its own Init operation 504 and Load operation 506 by virtue of deferred control object creation. The order of operation processing by the page object and descendent control objects depends on various factors, including without limitation the nature of the data in the HTTP request, the configuration of the control object hierarchy, the current state of the control objects, and whether deferred control

The Init operation 504 initializes a control object after it is created by executing any server-side code associated with initialization in the dynamic content resource. In this manner, each server-side control object may be customized with

specific server-side functionality that is declared in the dynamic content resource. In an embodiment of the present invention, dynamic content code intended to customize or extend the base page control classes as declared by the page developer in the ASP+ resource on the server. When the ASP+ resource is compiled, the declared code is included in the appropriate initialization code (e.g., the Init() methods of the page object and the descendent control objects). The Init operation 504 executes this code to customize or extend the page base class and the base classes for descendent control objects.

In an embodiment of the present invention, state management of the server-side control objects is supported in a Load operation 506 and a Save operation 516, which use a transportable state structure to accommodate the stateless model for client server systems by restoring server-side control objects to their previous state. In one embodiment, the state is communicated to and from the server in one or more hidden HTML fields of an HTTP request/response pair or in another transportable state structure, although other transportable state structures are contemplated within the scope of the present invention.

In a given sequence of requests and responses relating to the current page between a client and a server, the states of one or more control objects are recorded into a transportable state structure by the Save operation 516 after the processing of a previous request. In an embodiment of the present invention, additional state information is also included in the transportable state structure, including hierarchical information or control object identifiers to allow the server to associate a given state with the appropriate control object. In a subsequent HTTP request, the state information is returned to the server in the transportable state structure. The server extracts the state information from the received transportable state structure and loads the state data into the appropriate control objects within the control object hierarchy to restore each control object to its state as it existed prior to a previous HTTP response. After the current request is processed, the states of one or more server-side control objects are again recorded into the transportable state structure by the Save operation 516, and the transportable state structure is returned to the client in the next HTTP response.

As a result of the Load operation 506, each server-side control object is placed in a state consistent with its state prior to a previous HTTP response. For

example, if a text box control object includes a property value equaling "JDoe" prior to a previous HTTP response, the Load operation 506 restores the same control object to its previous state, in part by loading the text string "JDoe" into the property value. In addition, whether the state of a given object is stored and restored is configurable.

In summary of one embodiment of the present invention, the state of one or more server-side control objects is "saved" after processing. The saved state information is transmitted to the client in a response. The client returns the saved state information to the server in a subsequent response. The server loads the state information a freshly instantiated server-side control object hierarchy, such that the state of the hierarchy is restored to its previous state.

An alternative embodiment may maintain the state information on the server or at some other web location accessible by the server during the round trip from the server to the client, and then back to the server. After the client request is received by the server, this state information may be retrieved by the server and loaded into the appropriate server-side control object(s) in the control object hierarchy.

In operation 508, postback data received from the HTTP request is processed. Postback data may be included in the payload of the HTTP request in key-value pairs, in a hierarchical representation (e.g., XML), or in other data representations, such as RDF ("Resource Description Framework"). Operation 508 parses the payload to identify a unique identifier of a target server-side control object. If the identifier (e.g., "page:tablelist1:listrow2:label1") is found and the target server-side control object exists in the control object hierarchy, the corresponding postback data is passed to the control object. For example, referring to FIG. 1, a unique identifier and corresponding data (i.e., "12945") associated with a data item of list row 142 and column 144 is communicated in the payload of the HTTP request 114 to the web server 116. Operation 508 parses the payload of the HTTP request 114 and obtains the unique identifier of the table list 140 and its associated values (i.e., HTML code for the child user interface elements, such as list row 142). Operation 508 then resolves the unique identifier of the list row 142 to identify and traverse to the corresponding target server-side control object, passing a post back value to the target control object for processing.

As discussed with regard to the Load operation 506, the property values of server-side control objects may be restored to their previous states. In response to the receipt of postback data, the server-side control object determines whether the passed-in postback value causes a change from the corresponding property's previous value. If so, the change is logged in a change list to indicate a data change for the associated control object. After all postback data has been processed within the control object hierarchy, a call may be made to a control object method to raise one or more postback data changed events to one or more non-user-interface server components, such as a stock price look-up application running on the server. An example of a postback data changed event is an event indicating that postback data has caused a property of a server-side control object to change. In an exemplary embodiment, such an event can be sent to a system-provided event queue so that server-side code, which may be registered to process the event, can be invoked. The server-side code may then call a method of the non-user-interface server component. In this manner, a server-side non-user-interface server component can respond to events triggered by a change in data of a server-side control object. Alternative methods of implementing events are also contemplated in the scope of the present invention, including using application-provided event queues, polling, and processing interrupts.

In operation 510, postback events are handled. Postback events are communicated in the payload of the HTTP request. Operation 510 parses a specified event target (e.g., labeled “\_EVENTTARGET” in an embodiment of the present invention) identifying the server-side control object to which the event is directed. Furthermore, operation 510 parses the located event arguments, if any, and provides the event argument (e.g., labeled “\_EVENTARGUMENT” in an embodiment of the present invention) to the specified server-side control object. The control object raises its events for processing by a non-user-interface server component associated with the dynamic content resource.

Operation 512 resolves data binding relationships between the server-side control objects and one or more server-side databases accessible by the server, thereby updating in control object properties with database values and/or updating database fields with values of control object properties. In an embodiment of the present invention, properties of server-side control objects may be associated (or

data bound) to properties of a parent data binding container, such as a table in a server-side application database. During the data binding operation 512, the page framework may update a data bound control object property with the value of the corresponding parent data binding container property. In this manner, user interface elements on the web page of the next response accurately reflect updated property values, because the control object properties to which the user interface elements correspond have been automatically updated during the data binding operation 512. Likewise, control object properties can also be updated to the parent data binding container fields, thereby updating a server-side application database with postback input from a server-side control object.

A data binding relationship between a given server-control object and a given data source (e.g., a server-side dated table) is declared in the ASP+ resource (see, for example, line 14 in FIG. 6A). To support the commonly hierarchical nature of bound data in a web page, the data binding relationship may support a hierarchy of its own (see discussion of Fig. 7). In the control object hierarchy, the data binding relationships identify the data table and its property to which a property of a server-side control object is bound. A data binding relationship may also be bidirectional, meaning that: (1) an property value of a server-side data table may be loaded into the bound property of a server-side control object; and (2) a property value of a server-side control object may be loaded into the bound property of a server-side dated table. All server-side control objects accurately reflect appropriately updated data bound property values from a server-side dated table after the data binding operation 512.

Operation 514 performs miscellaneous update operations that may be executed before the control object state is saved and the output is rendered. Operation 516 requests state information (i.e., viewstate) from one or more control objects in the control object hierarchy and stores the state information for insertion into a transportable state structure that is communicated to the client in the HTTP response payload. For example, a "grid" control object may save a current index page of a list of values so that the "grid" control object may be restored to this state after a subsequent HTTP request (i.e., in operation 506). As described above, the viewstate information represents the state of the control object hierarchy prior to any subsequent actions by the client. When the viewstate information is returned, it will

be used to place the control object hierarchy in that previous state prior to processing any client postback input or databinding.

The render operation 518 generates the appropriate authoring language output (e.g., HTML data) for communication to the client in an HTTP response. Rendering is accomplished through a top-down hierarchical tree walk of all server-side control objects and embedded rendering code. Operation 520 performs any final cleanup work (e.g., closing files or database connections) before the control object hierarchy is terminated. Processing then returns to operation 502 and proceeds to operation 522, where the page object is terminated by calling its destructor.

FIG. 6A illustrates an exemplary dynamic content resource (e.g., an ASP+ resource) declaring data bound control objects on a server in an embodiment of the present invention. Line 1 includes the starting tag of an HTML file, <html>, as a literal. Lines 2-8 represent a code declaration block for code that populates local "MyOrders" and "MyItems" data arrays. Specifically, line 2 declares the server-side code as a server-side script by virtue of the "runat=server" attribute and value. The script overrides the Load operation 506 of FIG. 5 of the page object. Lines 3-5 declare public objects accessible within the name space of the server-side control objects. Line 3 declares a public object, "MyOrderSystem" of type "OrderSystem", which is used in line 7 to retrieve from a server-side datastore the data array that is declared in line 4 as the public data array object, "MyOrders". In line 8, the public object, "MyOrderSystem", is also used to retrieve from a server-side datastore a data array that is declared in line 5 as the public data array object, "MyItems".

Line 11 includes a literal that is the starting tag, <body>, of the body of an HTML file. Line 12 includes literal HTML code defining a header, "H1", for display on the web page. Line 13 includes a declaration of a server-side form control object. Line 14 declares a server-side table list control object having a data binding data source property equaling "Page.MyOrders". Therefore, the MyOrders data table is declared the data source for the table list control object and its descendants.

The in-line template declaration of lines 15-24 declares one or more child control objects of the table list control object declared in line 14. The template declaration is interpreted at parse time to dynamically bind an appropriate ITemplate



interface instance to a property of the same name in the iterating control object (e.g., the table list control object). The table list control object (i.e., an iterating control object) uses the ITemplate instance at runtime to create and initialize 0 to N new instances of the appropriate template control class. The number of new instances created may be controlled by the number of corresponding data sets (e.g., data rows) in the data source.

For example, lines 15-24 declare a template, called "itemtemplate", for the list row control objects of the table list control object declared in line 14. The parameters of the template control object are declared in lines 16-22, indicating a list row control object including an OrderID label control object, a Quantity label control object, and another table list control object. The table list control object declared in line 14 iterates through the data sets of the data table MyOrders to create a descendent list row control object for each data set.

The list row control objects corresponding to the itemtemplate template are developed by a control object developer as IBindingContainer instances and are therefore deemed "binding containers". During the load operation 506 of Fig. 5, the MyOrders data table is populated from the MyOrderSystem by the code in line 7. For example, the data provided by the MyOrdersSystem object may be extracted from a server-side database containing order and product data. Thereafter, the table list control object creates a list row control object, as a binding container, for each data set of the data table MyOrders.

The OrderID control object declared in line 16 is bound to the OrderID property of data sets in the MyOrders data table. A hierarchical data binding relationship between the OrderID label control object and the list row control object, defined by a PropertyBinding object of the OrderID label control object, establishes that the Text property of the label control object declared in line 20 is data bound to a data item property identified as "OrderID". The hierarchical data binding relationship also specifies that the data item property is a property of control object's binding container, namely the MyOrders data table. Although the property of the MyOrders data table that is bound to the OrderID control object is a data item, any public properties of a server side data array (e.g., a data table or another type of data object) may be bound to a property of a server-side control object, including a row index, a data value size, data value type, etc.

The Quantity control object declared in line 17 is bound to the Quantity property of data sets in the MyOrders data table. A hierarchical data binding relationship between the Quantity label control object and the list row control object, defined by a PropertyBinding object of the Quantity label control object, establishes that the Text property of the label control object declared in line 17 is data bound to a data item property identified as "Quantity". The hierarchical data binding relationship also specifies that the data item property is a property of the Quantity label control object's binding container, namely the MyOrders data table. Although the property of the MyOrders data table that is bound to be Quantity label control object is a data item, any public properties of a server side data array (e.g., a data table or another type of data object) may be bound to a property of a server-side control object, including a data set index, a data value size, data value type, etc.

Line 18 includes a declaration of a server-side table list control object having a data binding data source property equaling "Page.MyItems". Therefore, the MyItems is declared the data source for the table list control object and its descendants. The in-line template declaration of lines 19-22 declares one or more child control objects of the table list control object declared in line 18. The table list control object (i.e., another iterating control object) uses a template instance associated with the in-line template declaration to create and initialize 0 to M new instances of the appropriate template control class. The number of new instances created may be controlled by the number of corresponding data sets in the data source.

The parameters of the template control object are defined in lines 20-21, indicating a list row control object including an Index label (i.e., corresponding to an ordinal index of a given data set) and an Item label displaying the text of a data item property from the data source. For each data set in the data table MyItems, a list row control object is created under the table list control object declared in line 18. The remaining lines in the file 600 including closing tags to close the nested declarations within the file.

The ASP+ resource 600 illustrates declarations and server-side scripting for implementing unidirectional data binding (i.e., sending data from a server-side data table to the control object hierarchy) in an embodiment of the present invention. In an alternative embodiment, unidirectional data binding in the opposite direction (i.e.,

sending data from the control object hierarchy to a server-side data table) or bidirectional data binding (i.e., sending data in both directions) may be declared.

To effect the first type of unidirectional binding (i.e., toward the control object), code on line 7 overrides the Load() subroutine, for example, to load a snapshot data array of the orders from the MyOrderSystem data table into the local MyOrders data array (i.e., the first type of unidirectional data binding discussed). Thereafter, during the data binding stage, server-side control object properties may be updated from values from the local MyOrders data array, in accordance with the data binding statements on lines 14, 16, 17, 18, 20, and 21.

To effect the second type of unidirectional data binding (i.e., from the control object), a page developer may override the Save() subroutine of the page object (see Save operation 516 of FIG. 5). Previously, during the data binding stage, local MyOrders and MyItems data arrays may have been updated with values from server-side control object properties, in accordance with the data binding statements on lines 14, 16, 17, 18, 20, and 21. Thereafter, the overridden Save() subroutine loads the values of the local data array into the data tables of the MyOrderSystem object. An exemplary code declaration block overriding the Save() subroutine of the page object is shown in lines 6-9 in the ASP+ resource 602 of FIG. 6B. This type of unidirectional binding may be used, for example, to load orders into a server-side database. To declare bidirectional data binding, a page developer may override both the Load() and Save() subroutines of the page object.

FIG. 7 illustrates a partial control object hierarchy resulting from the processing of the exemplary dynamic content resource (e.g., an ASP+ resource) of FIG. 6A in an embodiment of the present invention. The top-level page control object and various literal control objects are omitted from the control object hierarchy 700. A form control object 702, corresponding to the form declaration starting at line 11 of FIG. 6A, contains a table list control object 704. The table list control object 704 is declared starting at line 12 of FIG. 6A to have a data binding property equaling the MyOrders data table.

The table list control object 704 contains list row control objects 706-708 (i.e., ListRow0 through ListRowN). The actual number of list row control objects at this level of the control object hierarchy is dependent upon the number of data sets in the MyOrders data table. The configuration of each list row control object is

defined by the itemtemplate declaration starting at line 13 of FIG. 6A. The data binding relationship of each list row control object is defined relative to the data source of each list row control object's binding container (i.e., the table list control object 704). In one embodiment, a control object binding container is the nearest higher level control object in the control object hierarchy that supports the IBindingContainer interface (see

A label control object 710 and a label control object 712, corresponding to the declarations on line 14 and 15 of FIG. 6A, are contained by the binding container list row control object 706. The data binding relationships of the label control objects 710 and 712 are defined relative to the binding container list row control object 706, which is data bound to the data sets of the MyOrders data table.

A table list control object 714, corresponding declaration on line 16 of FIG. 6A, is also contained by the binding container list row control object 706. The table list control object 714 is a binding container that is declared as a data bound to the sub-data table MyOrders.MyIngredients. The table list control object 714 contains list row control objects 716-718 (ListRow0 through ListRowN), corresponding to the data sets of the sub-data table MyOrders.MyIngredients. Each list row control object 716-718 is defined by the template declaration starting at line 19 of FIG. 6A. The data binding relationships of the list row control objects 716-718 are defined relative to the binding container table list control object 714.

Each list row control object 716-718 contains label control objects (e.g., 720 and 722). For example, a label control object 720 is declared on line 20 of FIG. 6A as data bound to the index field of a data set of the sub-data table MyOrders.MyIngredients, and a label control object 722 is declared on line 21 of FIG. 6A as data bound to an Ingredients data item of the sub-data table MyOrders.MyIngredients. To determine the data source of the Ingredients data item, properties of the label control object 720 and 722 return a reference to the nearest binding container in a higher level of hierarchy (i.e., a control object that is an IBindingContainer instance, such as the list row control object 716).

The hierarchy under the list row control object 708 follows the same configuration as the hierarchy under the list row control object 706 because it is defined by the same template declarations. The hierarchy includes a first level comprising label control object 730, label control object 734, and table list control

object 732. Beneath the table list control object 732, in other level of hierarchy comprises 0 to N list row control objects 736-738. Each list row control object 736-738 contains to label control objects (see label control objects 742, 744, 746, and 748) as defined by the corresponding template declaration.

FIG. 8 illustrates a representation of an exemplary server-side Control class in an embodiment of the present invention. The server-side Control class defines the methods, properties and events common to all server-side control objects in an embodiment of the present invention. More specific Control classes (e.g., a server-side button control object that corresponds to a client-side button in a web page) are derived from the control class. A Control class 800 is illustrated as including memory storing properties 802 and methods 804. Other Control class embodiments having a different combination of data members and methods are also contemplated within the scope of the present invention.

In the illustrated embodiment, properties 802 and methods 804 are public. Property "ID" is a readable and writable string value indicating a control object identifier. Property "Visible" is a readable and writable Boolean value that indicates whether the authoring language data for a corresponding client-side user interface element should be rendered. Property "MaintainState" is a readable and writable Boolean value that indicates whether the control object should save its viewstate (and the viewstate of its children) at the end of the current page request (i.e., in response to a save operation 516 in FIG. 5). Property "Parent" is a readable reference to a ControlContainer object associated with the current control object in the control object hierarchy. Property "Page" is a readable reference to the root page object in which the current control object is hosted. Property "Form" is a readable reference to a FormControl object in which the current control object is hosted. Property "Trace" is a readable reference to a TraceContext that allows a developer to write a trace log. Property BindingContainer is a readable reference to the control object's immediate data binding container. Property "Bindings" is a readable reference to a collection of the control object's data binding associations.

Methods 804 include methods for processing requests and accessing data members of the control object. In one embodiment, the methods are referenced by pointers stored in the memory space of the Control object. This referencing technique is also employed in embodiments of other server-side objects, including a

container control object, a control collection object, and a page object. Method "Init()" is used to initialize child control objects after they are created (see operation 504 of FIG. 5). Method "Load()" is used to restore the viewstate information from a previous HTTP request (see operation 506 of FIG. 5). Method "Save()" is used to save viewstate information for use with a later HTTP request (see operation 516 of FIG. 5). Method "PreRender()" is used to perform any pre-rendering steps necessary prior to saving viewstate and rendering content (see operation 514 of FIG. 5). Method "Render (TextWriter output)" is used to output authoring language code for the user interface element corresponding to the current control object (see operation 518 of FIG. 5). The code is output to the TextWriter Output Screen passed to it in the "Output" parameter. Method "Dispose()" is used to perform final clean-up work before terminating the control object (see operation 520 of FIG. 5).

Method "GetUniqueID()" obtains a unique, hierarchically qualified, string identifier for the current control object. Method "GetControlWithID(String id)" returns a reference to an immediate child control object with the provided identifier ("id"). Method "GetControlWithUniqueID(String id)" returns a reference to a child control object having a unique hierarchical identifier ("id").

Method "PushControlPropertyTwoBindingContainer (String prop Name)", an exemplary push module, is used to update a binding container for two-way data binding from post-back data when the post-back data value changes within the server-side control object. Method "PushBindingContainerPropertyTwoControl(String prop Name)", another exemplary push module, is used to update a server-side control object property with a current binding container value. Method "HasBindings()" returns of Boolean value indicating whether a control object has any binding associations. These methods are used to resolve binding relationships between properties of server-side control objects and properties in server-side data arrays (see the data binding operation 512 of FIG. 5), which are associated with server-side datastores.

FIG. 9 illustrates a flow diagram of operations for data binding a property of a server-side control object to a property of a server-side data array in an embodiment of the present invention. The operations illustrated in FIG. 9 may execute in one or more of the operations illustrated in FIG. 5. For example, in an embodiment of the present invention, operations 900 and 902 execute during a

creation operation or the Init operation 504, and the operation 904 executes during the Load operation 506, which results in data table being stored as a property of the table list control object in operation 906. When the table list control object or its children are access during the postback input processing operation 508 and 510 or the databinding operation 512, operations 908, 910, 912, and 914 execute to extend the hierarchy to include the binding containers and their children, in accordance with the configuration of the data table. Operations 916, 918, 920, and 922 execute during the databinding operation 512 of FIG. 5, and operation 924 executes during the Save operation 516 of FIG. 5. The operational correspondence discussed above applies to one embodiment of the present invention; however, other embodiments may reorder or realign the operations of FIGs. 5 and 9.

Turning to the individual operations of FIG. 9, if the page supports unidirectional data binding from a server-side database or bidirectional data binding between the server-side database and the control object hierarchy, operation 900 loads a local data table (i.e., a data array) from a portion of the server-side database using functionality of COM+ reflection. Operation 902 associates the data table as the data source of a table list control object. In one embodiment, this association is accomplished by means of the data binding property specified in declaration of the table list control object (see line 14 of Fig. 6A). Operation 904 creates the table list control object in the control object hierarchy. Operation 906 stores the data table as a property of the table list control object.

In operation 908, the table list control object iterates over each data set of the data table. For each data set detected, the table list control object creates a binding container control object (e.g., a list row control object) based on an associated template declaration. Each binding container control object is treated as a child of the table list control object. Operation 910 associates each binding container control object with a data set of the data table by setting a data binding property of the binding container control object to equal a data set of the data source (e.g., "MyOrders.Row(1)").

In operation 912, each binding container control object creates a descendant control object corresponding to a property of the binding container's corresponding data set of the data source. Alternatively, each binding container control object creates a child control object, which eventually leads to the creation of the

descendant control object corresponding to a property of the binding container's corresponding data set of the data source. Operation 914 establishes a binding relationship between the properties of the descendant control objects in the properties of corresponding data sets of the data source. The binding relationships may be defined by a PropertyBinding object stored in a property of the descendant control object.

Decision operation 916 detects whether data from the control object property should be sent to the corresponding binding container property. If so, operation 918 calls the PushControlPropertyToBindingContainer() method in the descendant control object to send the data to the binding container. Thereafter, processing proceeds to decision operation 920. Alternatively, if decision operation 916 does not determine that the data from a control object property should be sent to the corresponding binding container property, processing also proceeds from decision operation 916 to decision operation 920, which determines whether data should be sent from a binding container property to the corresponding control object property. If so, operation 932 calls the PushBindingContainerPropertyToControl() method in the descendant control object to receive the data from the binding container. Thereafter, processing proceeds to operation 924. Alternatively, if decision operation 920 determines that the data from the binding container property should not be sent to the control object property, processing also proceeds from decision operation 920 to operation 924. If the page supports unidirectional data binding to a server-side database or bidirectional data binding between the server-side database and the control object hierarchy, operation 924 saves the local data table to a portion of the server-side datastore by way of COM+ reflection. Processing can then continue with the Render operation 518 of FIG. 5.

FIG. 10 illustrates a representation of an exemplary server-side PropertyBindingCollection class in an embodiment of the present invention. The server-side PropertyBindingCollection class defines the methods and properties of a PropertyBindingCollection object (see the Bindings property of the server-side Control class in FIG. 8). A PropertyBindingCollection class 1000 is illustrated as including memory storing properties 1000 into and methods 1004. Other PropertyBindingCollection class embodiments having a different combination of



data members and methods are also contemplated within the scope of the present invention.

In the illustrated embodiment, properties 1002 and methods 1004 are public. Property "All" is a readable and writable snapshot array of PropertyBinding objects representing all property bindings in the collection, ordered by an index. Property "this[int index]" is a readable PropertyBinding object that returns a reference to an ordinal-indexed PropertyBinding object within the collection. Property "Count" is a readable integer value indicating the number of PropertyBinding objects within the collection.

Methods 1004 include methods for accessing the data of the PropertyBindingCollection object. Method "Add(PropertyBinding value)" is used to add specified PropertyBinding objects to the collection. Method "IndexOf(PropertyBinding value)" is used to obtain an ordinal index of a PropertyBinding object in the collection. Method "GetEnumerator(bool AllowRemove)" is used to obtain an enumerator of all PropertyBindings within the collection. Method "Remove(PropertyBinding value)" is used to remove a specified PropertyBinding object from the collection. Method "Remove(int index)" is used to remove an index-specified PropertyBinding object from the collection. Method "Clear()" is used to remove all PropertyBinding objects from the collection.

FIG. 11 illustrates a representation of an exemplary server-side PropertyBinding class in an embodiment of the present invention. The server-side PropertyBinding class defines the methods and properties of a PropertyBinding object (see the "All" and "this[int index]" properties of the PropertyBindingCollection class of FIG. 11), which defines the data binding relationship between a control object and a server-side data source. A PropertyBinding class 1100 is illustrated as including memory storing properties 1102. Other class embodiments having a different combination of data members and methods are also contemplated within the scope of the present invention.

In the illustrated embodiment, properties 1102 are public. Property "BindingContainer" is a readable and writable BindingContainer interface (i.e., an IBindingContainer object) used to identify and gain access to a binding container data source. The IBindingContainer interface is a marker interface that identifies a control that exposes its public properties to the bindings of its descendants.

Therefore, the "BindingContainer" property provides a BindingContainer identifier that allows a bound control object to locate and access the server-side data source to which is bound. Property "Property" is a readable and writable string value (i.e., a property identifier) identifying the bound property within the BindingContainer object. Property "Control" is a readable and writable Control object (i.e., identifying the target control object to which a data source property is bound. Property "ControlProperty" is a readable and writable string value (i.e., a property identifier) identifying the target property of the target control object. Property "FormatString" is a readable and writable string value identifying an optional format convergence string.

The embodiments of the invention described herein are implemented as logical steps in one or more computer systems. The logical operations of the present invention are implemented (1) as a sequence of processor-implemented steps executing in one or more computer systems and (2) as interconnected machine modules within one or more computer systems. The implementation is a matter of choice, dependent on the performance requirements of the computer system implementing the invention. Accordingly, the logical operations making up the embodiments of the invention described herein are referred to variously as operations, steps, objects, or modules.

The above specification, examples and data provide a complete description of the processes, systems, computer program products of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

#### 4 Brief Description of Drawings

FIG. 1 illustrates a web server for dynamically generating web page content for display on a client in an embodiment of the present invention.

FIG. 2 illustrates a flow diagram of operations for processing and rendering client-side user interface elements using server-side control objects in an embodiment of the present invention.

FIG. 3 illustrates exemplary modules in a web server used in an embodiment of the present invention.

FIG. 4 illustrates an exemplary system useful for implementing an embodiment of the present invention.

FIG. 5 illustrates an operation flow diagram representing processing of a page object in an embodiment of the present invention.

FIG. 6A illustrates an exemplary dynamic content resource (e.g., an ASP+ resource) declaring data bound control objects in an embodiment of the present invention.

FIG. 6B illustrates an exemplary code declaration block declaring a second type of unidirectional data binding to a server-side dated table in an embodiment of present invention.

FIG. 7 illustrates a partial control object hierarchy resulting from the processing of the exemplary dynamic content resource (e.g., an ASP+ resource) of FIG. 6A in an embodiment of the present invention.

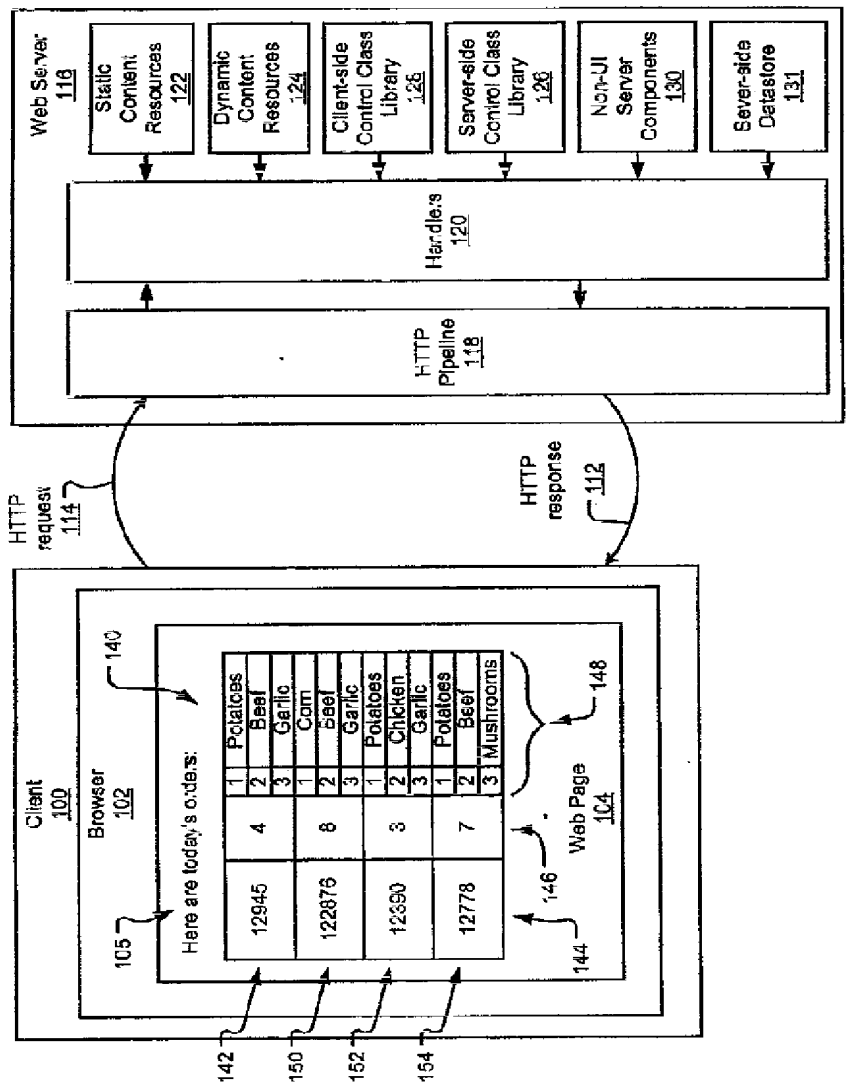
FIG. 8 illustrates a representation of an exemplary server-side Control class in an embodiment of the present invention.

FIG. 9 illustrates a flow diagram of operations for data binding a property of a server-side control object to a property of a server-side data array in an embodiment of the present invention.

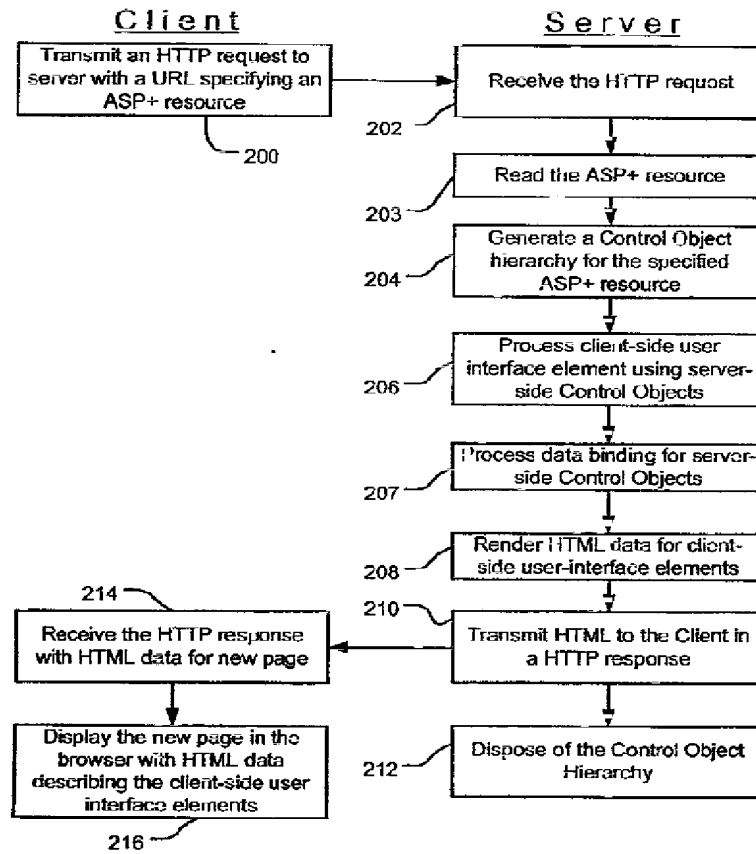
FIG. 10 illustrates a representation of an exemplary server-side PropertyBindingCollection class in an embodiment of the present invention.

FIG. 11 illustrates a representation of an exemplary server-side PropertyBinding class in an embodiment of the present invention.

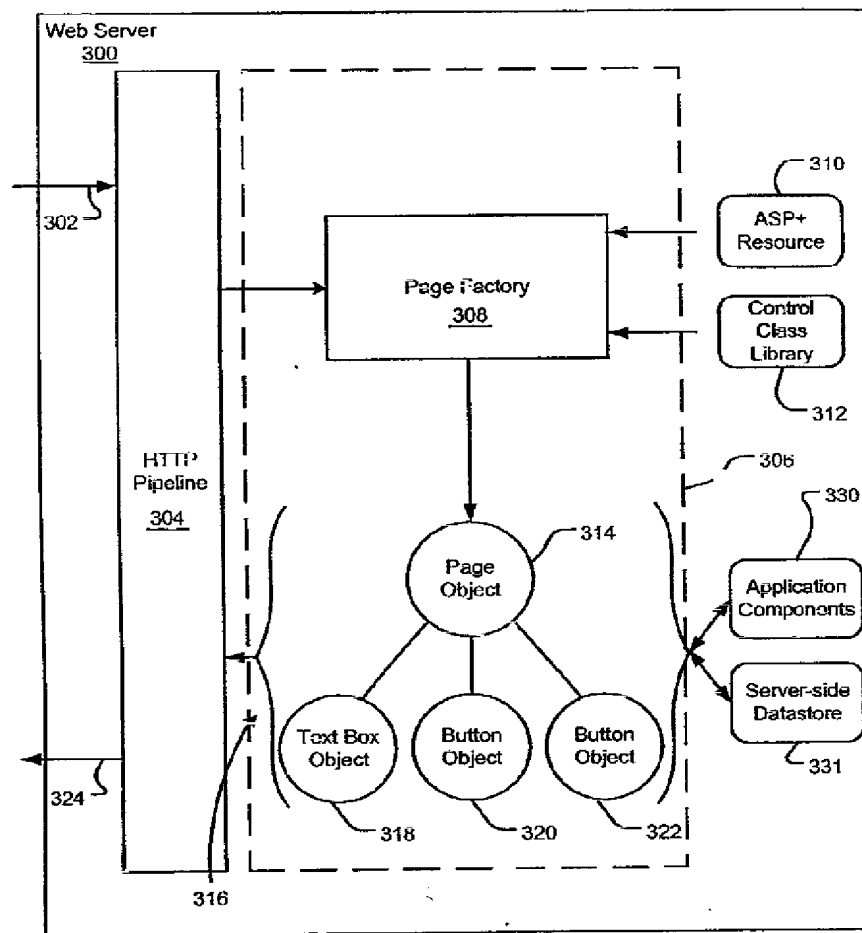
【図1】



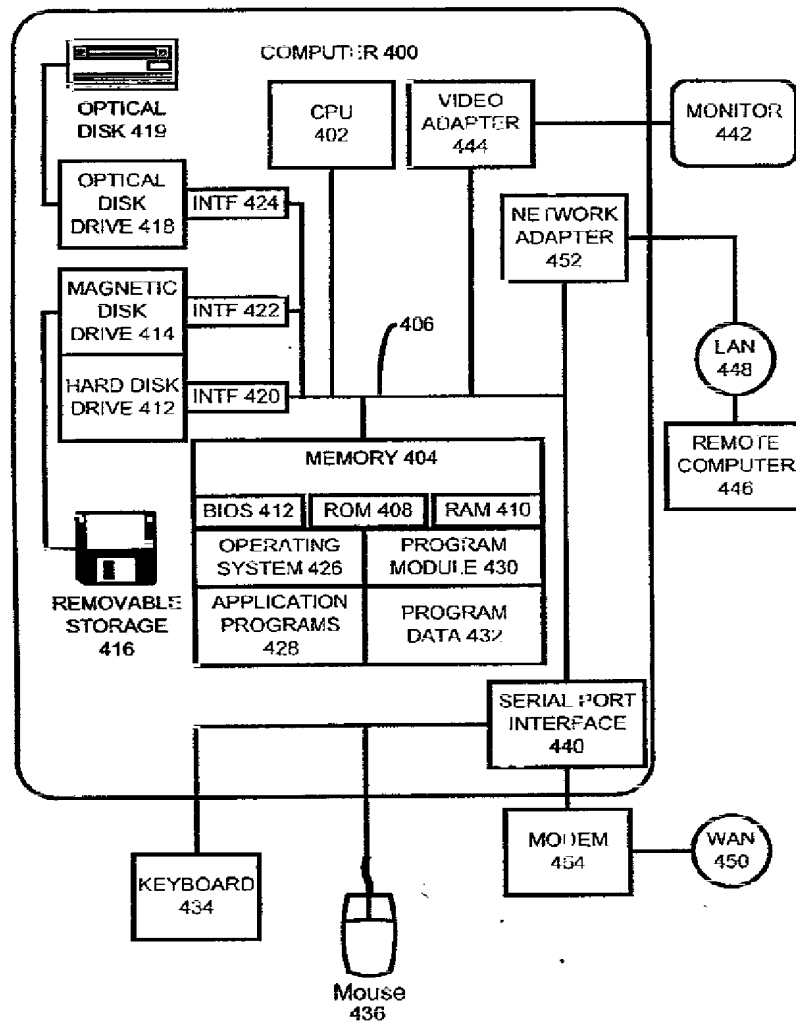
【図2】



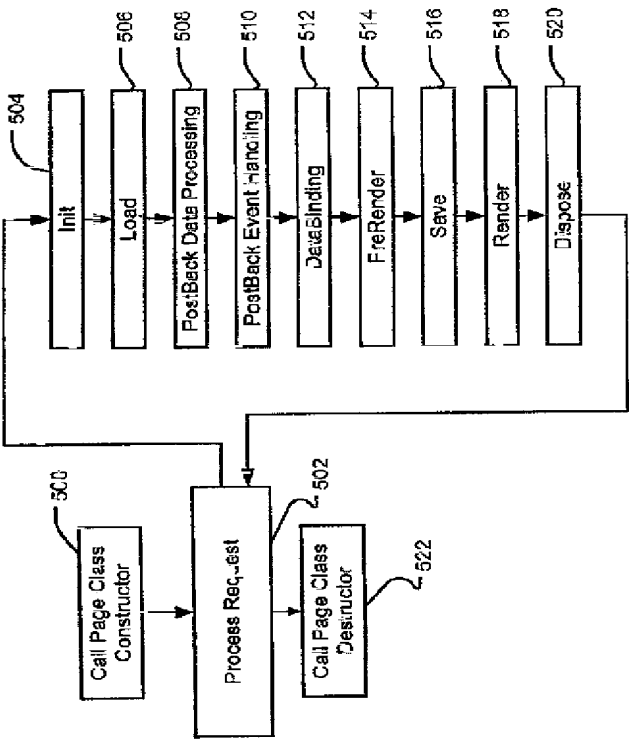
【図3】



【図4】



【図5】





【 6 A 】

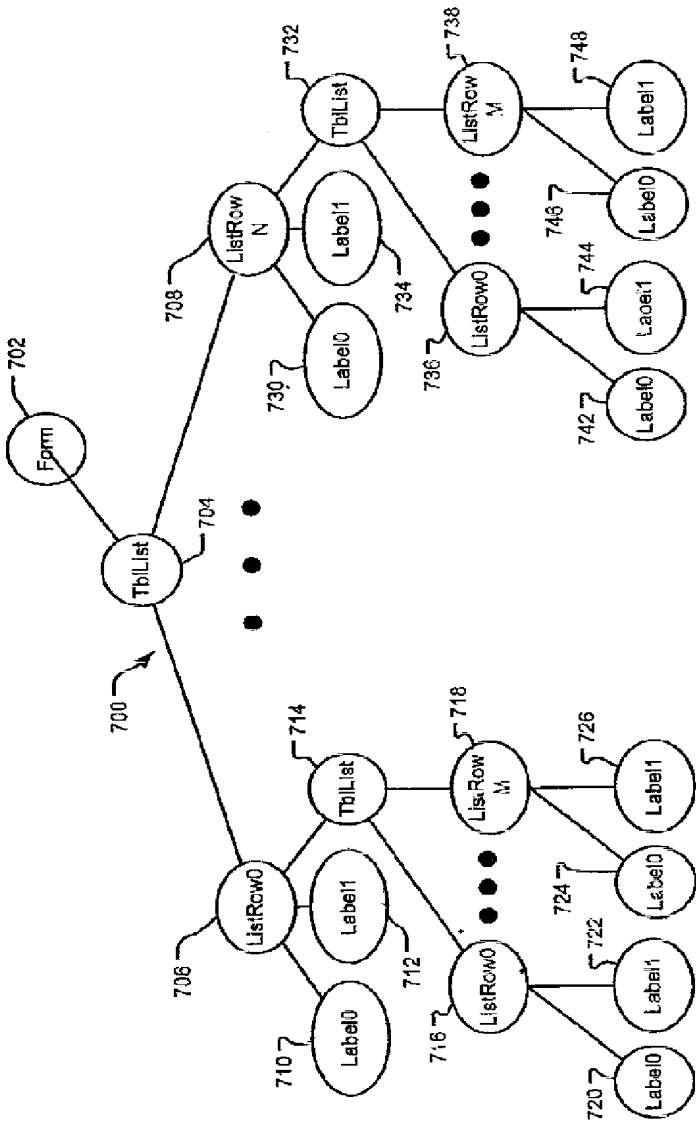
```

1  <html>
2  <script runat=server>
3      Public MyOrderSystem as New OrderSystem
4      Public MyOrders() as Order
5      Public MyIngredients() as Ingredients
6
7      Overrides Sub Load()
8          Set MyOrders = MyOrderSystem.GetOrdersFromDay(Now)
9          Set MyItems = MyOrderSystem.GetItemFromOrders(MyOrders, DataItem.OrderID)
10         End Sub
11     </script>
12
13     <body>
14         <h1> Here are today's orders: </h1>
15
16         <form runat=server>
17
18             <table data-binding="data-source:Page.MyOrders" runat=server>
19                 <thead>
20                     <tr>
21                         <th data-binding="Text:DataItem.OrderID" runat=server>
22                             <span data-binding="Text:DataItem.Quantity" runat=server>
23                                 <table data-binding="data-source:Page.MyOrders.MyIngredients" runat=server>
24                                     <thead>
25                                         <tr>
26                                             <th data-binding="Text:index" runat=server>
27                                                 <span data-binding="Text:DataItem.Item" runat=server>
28

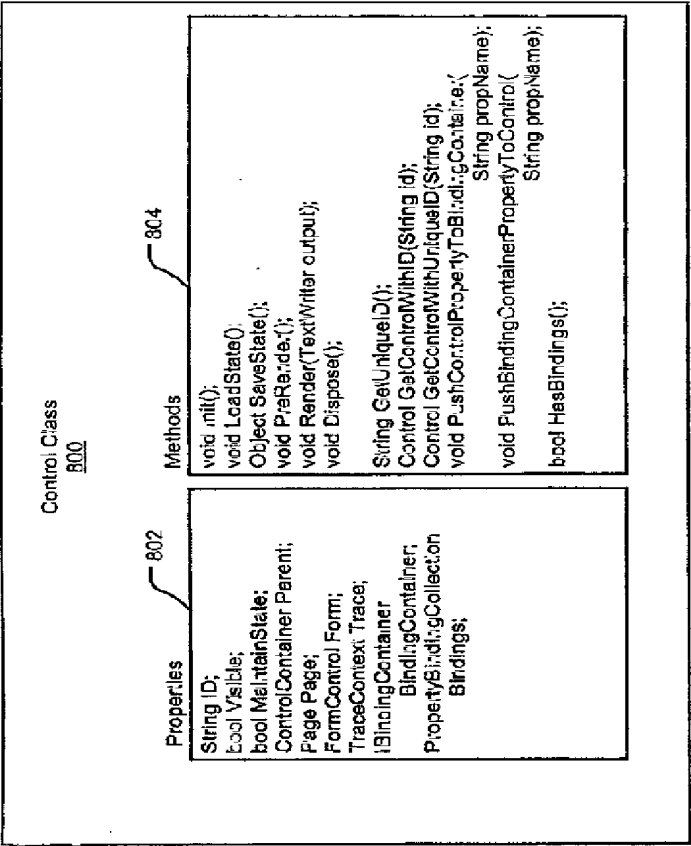
```



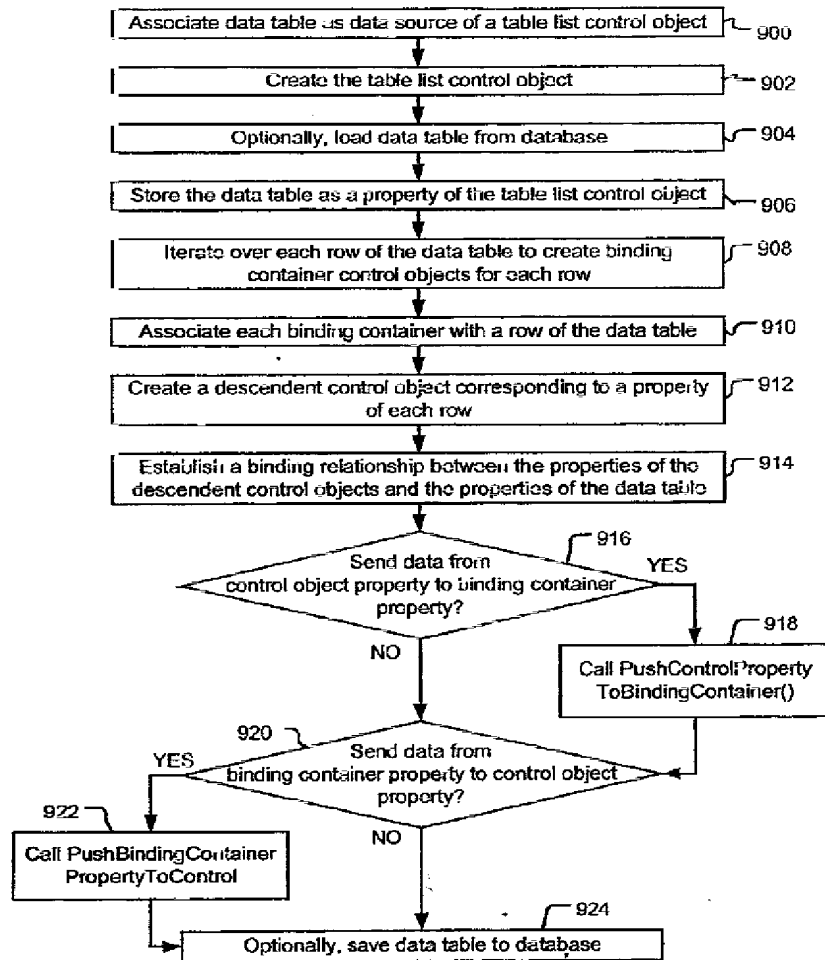
【図7】



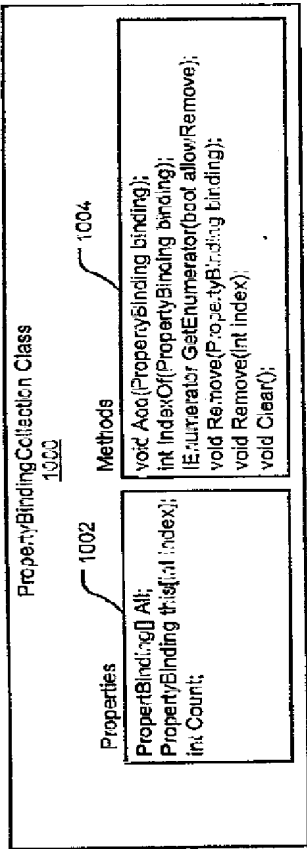
【図8】



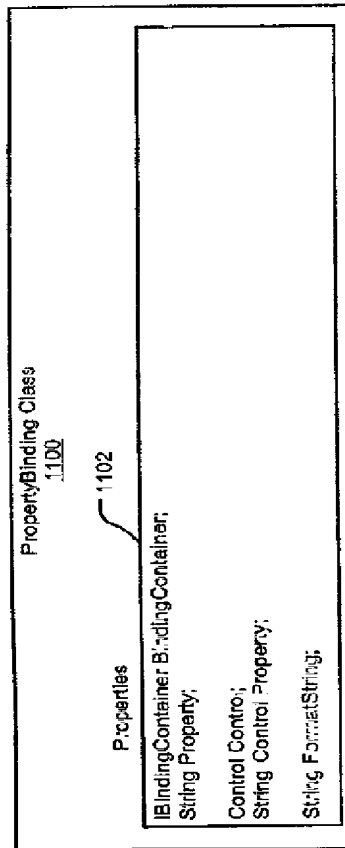
【 図 9 】



【 図 10 】



【図11】



## 1 Abstract

Server-side control objects corresponding to client-side user interface elements are created in a control object hierarchy. Properties of the control objects may be data bound to properties of a server-side data source (e.g., a server-side database). Hierarchical data binding relationships are established between properties of control objects and properties of a data source. Template declarations are used to define the configuration of binding container objects that correspond to data objects in the data source. An iterating control object determines the number of data objects in the data source increase according number of binding container objects. A simple data binding types include without limitation: (1) unidirectional data binding from the data source to a control object; (2) unidirectional data binding from a control object to the data source; and (3) bidirectional data binding between a control object and the data source.

## 2 Representative Drawing

Fig. 9